

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-202409

(43)Date of publication of application : 09.08.1996

(51)Int.Cl. G05B 15/02
G06F 15/16
G06F 15/16

(21)Application number : 07-012293

(71)Applicant : HITACHI LTD

(22)Date of filing : 30.01.1995

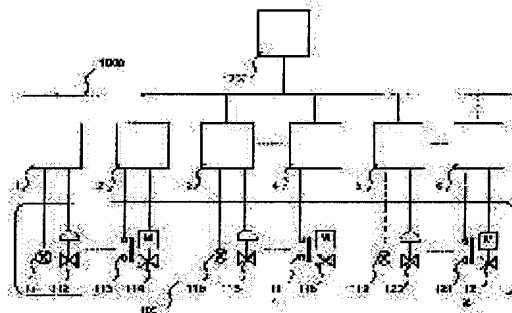
(72)Inventor : YAMAOKA HIROMASA
WATABE MITSURU
KASAHARA TAKAYASU
SHIBATA KATSUNARI
MATSUBARA KIYOSHI
MOROOKA YASUO
FUNABASHI SEIJU

(54) DECENTRALIZED CONTROL UNIT, SYSTEM, AND CONTROLLER

(57)Abstract:

PURPOSE: To provide a controller which has many kinds of various control function for a decentralized control system.

CONSTITUTION: Controllers 1-6 consist of plural processors which perform numerical operation, sequence operation, neural net(NN) operation, fuzzy operation, etc. The controllers 1-6 are reduced in size by the configuring technology with LSI. Further, processor functions and I/O functions are made programmable to increase flexibility. A program incorporates a simple standard program and learnt models of the NN, etc., and weights and adds plural calculated outputs to generate output data. Further, the value of weighting is updated by learning. Consequently, the decentralized control system having many kinds of various control function such as numerical operation control, sequence control, NN application control, and fuzzy application control and the controllers used for them can be provided.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-202409

(43) 公開日 平成8年(1996)8月9日

(51) Int.Cl. ⁵	識別記号	序内整理番号	F I	技術表示箇所
G 0 5 B 15/02				
G 0 6 F 15/16	3 7 0 Z			
	4 7 0 B			
		9063-3H	G 0 5 B 15/ 02	M
審査請求 未請求 請求項の数21 O L (全 50 頁)				

(21) 出願番号 特願平7-12293

(22) 出願日 平成7年(1995)1月30日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 山岡 弘昌

茨城県日立市大みか町五丁目2番1号 株式会社日立製作所大みか工場内

(72) 発明者 渡部 満

茨城県日立市大みか町七丁目1番1号 株式会社日立製作所日立研究所内

(72) 発明者 笠原 孝保

茨城県日立市大みか町七丁目2番1号 株式会社日立製作所エネルギー研究所内

(74) 代理人 弁理士 小川 勝男

最終頁に続く

(54) 【発明の名称】 分散制御装置、システム及びコントローラ

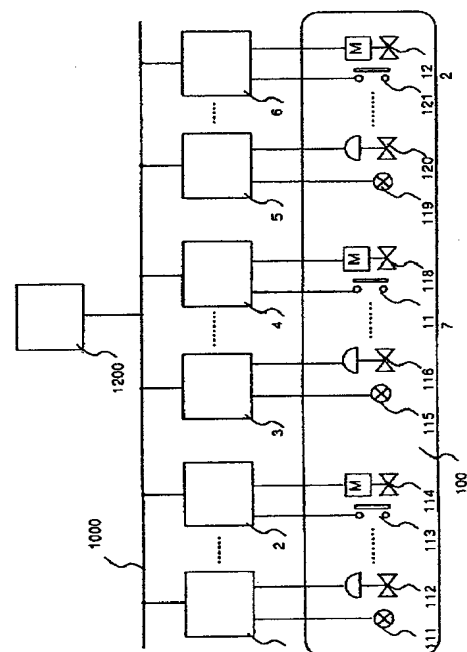
(57) 【要約】

【目的】 分散制御システムにおいて、多種多様な制御機能を有したコントローラを実現すること。

【構成】 コントローラを数値演算、シーケンス演算、ニューラルネット (NN)、ファジィなどを実行する複数のプロセッサで構成する。コントローラはLSI化技術により小型化する。さらに、プロセッサ機能、I/O機能をプログラマブルにすることにより柔軟性を増す。プログラムが簡単な標準プログラムとNN等の学習モデルを組み込んで、計算される複数の出力を、重み付けして加算することによって出力データを生成する。かつ該重み付けの値を学習によって更新していく。

【効果】 本発明により、数値演算制御、シーケンス制御、NN応用制御、ファジィ応用制御など多種多様な制御機能を有した分散制御システム及びそれに用いるコントローラを実現することができる。

図 1



【特許請求の範囲】

【請求項1】複数の制御対象機器を有するプロセスの制御装置において制御対象機器に制御指令を発するコントローラを各制御対象機器に対応して現場設置したことを特徴とする分散制御装置。

【請求項2】複数のプロセッサからなる制御システム用コントローラにおいて各コントローラ内の複数のプロセッサは自己の健全／異常の判別手段、プロセッサの負荷を測定または予測する手段を有し、一つのプロセッサが異常または負荷オーバーとなることを予測または検知した場合、他のプロセッサが負荷の代替を行うことを特徴とするコントローラ。

【請求項3】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて各コントローラのうち少なくとも一台のコントローラは複数のプロセッサから構成され、該コントローラ内の複数のプロセッサは自己の健全／異常の判別手段、プロセッサの負荷を測定または予測する手段を有し、一つのプロセッサが異常または負荷オーバーとなることを予測または検知した場合、同一コントローラ内の他のプロセッサが負荷の代替を行うことを特徴とする分散制御システム。

【請求項4】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて各コントローラのうち少なくとも一台のコントローラはメモリまたは入出力を共有する複数のプロセッサから構成され、該伝送手段は該コントローラ内プロセッサ群の正常、異常に関係なく該コントローラの上記メモリまたは入出力の内容を読み書きできるように構成したことを特徴とする分散制御システム。

【請求項5】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、上記プロセッサ群は上記伝送手段を制御する通信プロセッサを含み、該通信プロセッサが故障時は他のプロセッサが通信プロセッサを代替するように構成したことを特徴とする分散制御システム。

【請求項6】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、上記プロセッサのうち一つのプロセッサが故障時は他のプロセッサが該故障プロセッサを代替するように構成したことを特徴とする分散制御システム。

【請求項7】複数のプロセッサからなる制御システム用コントローラにおいて、各プロセッサの故障を検出する第1の手段(6180)と、各プロセッサの負荷状況を記憶する第2の手段(6160)と、各プロセッサの負荷を制御する第3の手段(6170)を有し、第1の手段(6180)が一つのプロセッサの故障を検出した場合に、第2の手段(6160)に記憶された各プロセッサの負荷状況に応じて第3の手段(6170)が故障し

たプロセッサの負荷を他の一つ以上のプロセッサに対して分割して代替させることを特徴とするコントローラ。

【請求項8】複数のプロセッサを有し、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの故障を検出する第1の手段(6180)と、各プロセッサの負荷状況を記憶する第2の手段(6160)と、各プロセッサの負荷を制御する第3の手段(6170)を有し、第1の手段(6180)が一つのプロセッサの故障を検出した場合に、第2の手段(6160)に記憶された各プロセッサの負荷状況に応じて第3の手段(6170)が故障したプロセッサのタスクを他の一つ以上のプロセッサに対して選択して代替させることを特徴とするコントローラ。

【請求項9】複数のプロセッサからなり、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの故障を検出する第1の手段(6180)と、各プロセッサの負荷状況を記憶する第2の手段(6160)と、各プロセッサの負荷を制御する第3の手段(6170)を有し、第1の手段(6180)が一つのプロセッサの故障を検出した場合に、第2の手段(6160)に記憶された各プロセッサの負荷状況に基づいて、第3の手段(6170)が故障したプロセッサのタスクと他の一つ以上のプロセッサが実行しているタスクとからその一つ以上のプロセッサの各々が実行するタスクを選択し、それらプロセッサが選択されたタスクを実行することを特徴とするコントローラ。

【請求項10】複数のプロセッサを有する制御システム用コントローラにおいて、各プロセッサの負荷を検出する第1の手段(6140)と、各プロセッサの負荷状況を記憶する第2の手段(6160)と、各プロセッサの負荷を制御する第3の手段(6170)を有し、第1の手段(6140)が一つのプロセッサの負荷オーバを検出した場合に、第2の手段(6160)に記憶された各プロセッサの負荷状況に応じて第3の手段(6170)が負荷オーバとなったプロセッサの負荷を一つ以上のプロセッサに対して分割して代替させることを特徴とするコントローラ。

【請求項11】複数のプロセッサからなり、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの負荷を検出する第1の手段(6140)と、各プロセッサの負荷状況を記憶する第2の手段(6160)と、各プロセッサの負荷を制御する第3の手段(6170)を有し、第1の手段(6140)が一つのプロセッサの負荷オーバを検出した場合に、第2の手段(6160)に記憶された各プロセッサの負荷状況に応じて第3の手段(6170)が負荷オーバとなったプロセッサのタスクを一つ以上のプロセッサに対して選択して代替させることを特徴とするコントローラ。

【請求項12】複数のプロセッサを有し、各プロセッサ

10

20

30

40

50

が複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの負荷を検出する第1の手段(6140)と、各プロセッサの負荷状況を記憶する第2の手段(6160)と、各プロセッサの負荷を制御する第3の手段(6170)を有し、第1の手段(6140)が一つのプロセッサの負荷オーバを検出した場合に、第2の手段(6160)に記憶された各プロセッサの負荷状況に基づいて、第3の手段(6170)が負荷オーバとなったプロセッサのタスクと他の一つ以上のプロセッサが実行しているタスクとからその一つ以上のプロセッサの各々が実行するタスクを選択し、それらプロセッサが選択されたタスクを実行することを特徴とするコントローラ。

【請求項13】複数のコントローラと各コントローラを接続するネットワークを有する分散制御システムにおいて、各コントローラは自コントローラの負荷を測定し測定した負荷をネットワークに送信する第1の手段(6140)と、任意のコントローラの負荷の代替を他のコントローラに依頼する第2の手段(6120)と、他のコントローラから代替の依頼に対して自己の負荷状況に基づき回答する手段をする第3の手段を有し、何れかのコントローラが負荷オーバあるいは故障した場合、その負荷オーバあるいは故障を検出したコントローラの第2の手段は各第1の手段から伝えられた負荷状況に応じて負荷の代替を依頼するコントローラを定め負荷の代替を依頼し、依頼されたコントローラの第3の手段は自己の負荷状況に応じて代替する負荷を定めこの負荷の実行を制御すると共にこの代替状況を回答することを特徴とした分散制御システム。

【請求項14】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて少なくとも一台のコントローラはグローバルメモリまたは入出力を共有する複数のプロセッサから構成され、上記プロセッサはキャッシュメモリおよび該キャッシュメモリの内容とグローバルメモリの内容一致化手段を有し、グローバルメモリの内容または入出力の内容は上記伝送手段を介して他のコントローラから読み取り可能なように構成したことを特徴とする分散制御システム。

【請求項15】複数のコントローラと各コントローラを接続する伝送手段からなる分散制御システムにおいて少なくとも一台以上のコントローラは入出力有し、該入出力の接続がプログラマブルであることを特徴とする分散制御システム。

【請求項16】複数のコントローラを有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、該複数のプロセッサ各々は自己の健全/異常の判別手段、各プロセッサの負荷を測定する手段を有し、一つのプロセッサが負荷オーバーになると、他のプロセッサのうち最も負荷の小さいプロセッサが負荷の代替を行うことを特徴とする分散制御システム。

ム。

【請求項17】複数のプロセッサを有する制御システム用コントローラにおいて、該複数のプロセッサのうち少なくとも一つのプロセッサはニューロ演算、ファジィ演算、数値演算またはシーケンス演算を実行することを特徴とするコントローラ。

【請求項18】それぞれが情報を格納するメモリを有する複数のプロセッサと上記各メモリ、各プロセッサ及び外部のネットワークとの間で上記メモリの情報を授受するために上記相互を連絡する伝送線と少なくとも一つのプロセッサにネットワークと各メモリ間の情報授受を制御するネットワーク制御機能を設定する手段とを有し、上記手段により特定された一つのプロセッサによって、上記ネットワークと上記メモリ間の情報授受を制御するようにしたことを特徴とするコントローラ。

【請求項19】センサ等からの入力を用いて制御出力を求める制御システムにおいて、該入力から少なくとも1つのニューラルネットワークを含む複数の手段または方法によって計算される複数の出力を、重み付けして総合することによって制御出力を生成し、かつ該重み付けの値を学習によって更新していくことを特徴とする適応的制御システム。

【請求項20】複数のコントローラをネットワークで接続して制御を実行する分散制御システムにおいて、該複数のコントローラにおいて感知した異常をネットワーク上に送信するか否かの判定をネットワークからの受信情報に基づいて各コントローラで実行する分散型コントローラの診断方式。

【請求項21】それぞれの機能及び相互接続がプログラマブルなプロセッサを複数内蔵するシングルチップコントローラ。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は各種プロセス制御、プラント制御、車両制御等において複数の制御装置(コントローラ)を用いて分散制御を行う分散制御システムに関する。

【0002】

【従来の技術】従来からプラント制御システムの例では、複数のコントローラを分散して制御を行う分散制御システム、DCS(Distributed Control System)と呼ばれる制御システムがある。このDCSは複数のコントローラとマンマシンシステムとしてのオペレータズコンソールをネットワークで結びプラントを制御しようとするものである。しかし、分散システムといってもコントローラ自体は計器室、電気室あるいは計算機室等と呼ばれる専用の設置スペースに集中配置し、そこからプラントのセンサあるいはアクチュエータへ入出力ケーブルを接続するのが常であった。しかし、この方法ではコントローラの専用設置スペースが必要になることと、コン

トローラの入出力とプラント間の接続ケーブルの敷設に費用がかかるという問題点があった。

【0003】また、従来のコントローラを分散配置するとコントローラが故障した際にその保守がすぐにはできないという問題点が発生する。

【0004】また、一般には一つのプラントは複数の制御対象グループから構成され、さらに一つの制御対象グループは複数の制御対象機器から構成されている。そこで、一つのコントローラでどれだけの制御対象機器を掌握するかが従来から議論されている。一般には、一台のコントローラで掌握する制御対象機器が少なければ少ないほど、一台のコントローラが故障してもその影響範囲は小さくなり危険分散化が図れるが、コントローラ間の通信量は増大するという問題があった。

【0005】一方で、制御の高度化は今後ますます進み、リレー接点ロジックの代替であるシーケンス制御や単なる数値演算制御だけではなく、多変数制御を行うための高速行列演算、あいまい量を取り扱うファジィ制御、学習能力をもつニューラルネットワーク応用制御等各種の制御が複合して用いられるようになってきている。しかし、このような制御システムは、各々専用の機能を持ったコントローラを適当に組み合わせて構築しているのが現状である。その結果、高度な制御を行なおうとすれば、分散システム内に多種類のコントローラを配置する必要があり、設計、プログラミング、運転、保守いずれの時点においても複雑さが増大しそれに伴うコストも増大するという問題があった。

【0006】また、一つのコントローラを複数のプロセッサで実現する所謂マルチプロセッサ構成のコントローラも実現されている。例えば、伝送処理を行うネットワークコントロールプロセッサ、数値演算プロセッサおよび高速シーケンスプロセッサというように複数のプロセッサでもってコントローラを実現するわけである。しかし、従来の方法では各種制御内容に応じた複数種類のプロセッサを用意し、制御対象に応じてその組み合わせを変える必要があった。即ち、各種制御対象のセンサー、アクチュエータの種類、点数、制御内容が異なるため、各プロセッサおよびプロセス入出力カードがプラグインユニット式で増設、交換できるように作られている。ところが、このコントローラを1ボード化またはLSI化しようすると非常に困難な事態が発生する。すなわち、制御対象毎に異なる機能のボードやLSIを作ることになり、開発期間、開発コスト面において大きな障害となるとともに保守面においてもすべての品種に対応した保守部品、保守管理が必要になり、保守コストの増大につながる。特にLSI化のメリットの1つは大量生産による低価格化にあり、この様に個別に多くの種類のLSIを開発することは、このLSIの最大のメリットを享受できないという事になる。

【0007】また、コントローラのプログラミングに関

しては、従来各制御部のプログラムをユーザが逐一書いていた。一般的には、より良い制御性能を得ようとするプログラムは複雑化する。また、従来のプログラムでは、経時変化や環境の変化、ユーザの好みに柔軟に対応していくことが大変困難であった。そこで、ニューラルネットなどを使用して徐々に学習しながら制御性を良くするというアプローチがある。しかし、従来の学習によるアプローチでは所望の出力を得るまでに、非常に多くの学習、経験を積み重ねなければならず、学習の初期においては所望とする制御出力とかけ離れたものしか得られないのが常である。一方、制御の初期段階から所望とする制御出力に近い値を得なければならないことも事実である。また、ユーザが持っている知識が優れている場合にはそれを最大限利用するといった方法が望れているが、ニューロ等による学習結果とユーザの知識のどちらが優れているかの判断が難しい。

【0008】さらに、制御性能の評価については、例えば、自動車の場合のように、その基準は乗り心地、スピード、温度など、複数存在することが多い。また、その価値基準は人によって、また、同じ人でも時によって異なる。このようなことに、柔軟に対応するためには、動作中にユーザがその価値基準算出法を変更可能であることが必要である。

【0009】ネットワークについては制御対象が定常状態であれば通信情報量の増減はあまり無いが、過渡状態特に異常発生時などは通信情報量が増加してしまう。例えば、従来の分散制御システムの診断方法であるブラックボードアーキテクチャによってもコモンモード故障時に個々のコントローラから通信ネットワーク上に多数の故障情報が送信されネットワークの通信ネックが発生する危険性があった。

【0010】またさらに、分散制御システムの稼働率と信頼性の向上に関しては、1台のコントローラが故障(ダウン)した場合に他のコントローラがその負荷を代替(バックアップ)する方法が一般的である。例えば、特開昭62-177634号と特開昭63-118860号に示されるように、あるコントローラのダウンに対してバックアップするコントローラをあらかじめ決めておく方法が知られている。さらに、特開平2-224169号と特開平3-296851号に示されるように計算機システムでは各計算機の負荷状況に応じて、ダウンした計算機の負荷を分割し複数の計算機でバックアップする方式が知られている。ところが、前者の方法はバックアップ専用のコントローラを準備する必要があった。また、後者の方法は負荷の分散を集中管理しているためその集中管理用の計算機がダウンした場合、計算機システムの全体が停止する可能性があった。

【0011】

【発明が解決しようとする課題】本発明が解決しようとする一つの課題は、一台のコントローラで掌握する制御

対象機器を最小限度に少なくし、危険分散を図ることが可能なコントロールシステムを実現することである。

【0012】またさらに、本発明が解決しようとするもう一つの課題は、コントローラの専用設置スペースおよびコントローラの入出力とプラント間の接続ケーブルの敷設を不要または最小にできる分散設置形のコントロールシステムにも適用できるコントローラを実現することである。

【0013】またさらに、本発明が解決しようとするもう一つの課題は、できる限り少ない種類で各種の制御対象、制御方式に対応可能な汎用性のあるコントローラを提供することである。

【0014】またさらに、本発明が解決しようとするもう一つの課題は小型かつ低コストなコントローラを提供することである。

【0015】またさらに、本発明が解決しようとするもう一つの課題は、小規模のコントローラから大規模のコントローラまで同一アーキテクチャで構成できるスケラビリティを有したコントローラを提供することにある。

【0016】またさらに、本発明が解決しようとするもう一つの課題は、ハードウェア完成後に目的に合わせて機能のプログラムが可能なコントローラを提供することにある。ここで言う機能のプログラムとは、数値演算機能、ファジィ推論機能、ニューラルネット処理機能、シーケンス演算機能などの専用処理機能をコントローラ内のハードウェアに埋め込むことである。

【0017】またさらに、本発明が解決しようとするもう一つの課題は、部分故障に対して自己修復可能なコントローラを提供することにある。

【0018】またさらに、本発明が解決しようとするもう一つの課題は、プログラミングの簡単なコントロールシステムを提供することにある。

【0019】またさらに、本発明が解決しようとするもう一つの課題は、コモンモード故障時に個々のコントローラから通信ネットワーク上に送信される故障情報を減じ、ネットワークの通信ネックによるシステムダウンが発生しにくい分散制御システムを提供することにある。

【0020】またさらに、本発明が解決しようとするもう一つの課題は、集中管理用のコントローラと待機用コントローラを設けずに、正常に動作しているコントローラがダウンしたコントローラの負荷を自律的に分割してバックアップすることが可能な分散制御システムを実現することである。特に、コントローラの多重ダウンに対しても、正常に動作しているコントローラが協調して、その余剰性能を有効に活用し分散制御システムの総合性能の限界までバックアップを実施することにある。

【0021】

【課題を解決するための手段】上記課題を解決するために、本発明では、コントローラの内部構成、外観構造と

もに拡張性のあるものとし、その基本単位をLSI化し小型のコントローラを実現できるようにした。従って、分散設置型としても集中設置型としても利用できる。また、本コントローラを制御対象機器単位に配置することにより、危険分散システムが実現できる。

【0022】さらに、汎用性をもったコントローラを実現するために、一つのコントローラを数値演算機能はもとよりニューロ、ファジィ、行列等を高速に演算するための並列演算機能を実行するプロセッサ、シーケンス制御演算を高速に実行するプロセッサ、さらには、ネットワークとの通信機能を実行するプロセッサ等の複数のプロセッサから構成し、各機能を並列に実行可能とした。また、入出力及びプロセッサを機能プログラムできるチップを実現する方法も提供した。

【0023】また、同一コントローラ内の一つのプロセッサがダウンしても他のプロセッサがその機能を代替できる手段を設けた。

【0024】制御演算部にニューラルネットと従来のようなユーザが付与したモデルを計算する手段または方法を設け、両者を重み付けして加えることによって制御出力を生成することとした。そして、その重み付けの値を学習が進んでいない状態ではユーザの付与モデル出力を100%またはそれに近い値とし、その後学習によって更新し、また、ニューロや他の手段からの出力値に対する重み付けの値もニューラルネットで行い、学習させることとした。

【0025】また、評価についても、状態の評価値をユーザに表示し、それをユーザが修正する手段を有し、さらに、新しい評価基準をユーザが定義すると共に、複数の評価値を何らかの加工をした後に最小値を求め、これを用いて出力の演算法を更新していくこととした。

【0026】さらに、故障診断時のネットワークの負荷を低減する手段としては、ネットワークの伝達情報をマンマシン装置に格納し、編集・処理するネットワークモニタ、および各コントローラにつけた入出力情報フィルタを設け冗長な情報はネットワークには乗らないようにした。

【0027】さらに、ダウンしたコントローラの負荷を自律的に分割してバックアップするための手段は、各コントローラが実行しているタスクとその負荷状況を示す実行分担表と、各タスクを実行したときの負荷を示すタスク負荷表と、各コントローラの負荷状況によりどのコントローラが余裕がありバックアップを実行しやすいかを示す受付順位表とに基づきバックアップを依頼するバックアップ依頼手段と、前記表によりバックアップを受け付け可能か判定し受け付けたか否かを回答するとともに受け付けたタスクの実行を指示するバックアップ受付手段とから構成される。

【0028】また、一つのコントローラ自身の耐故障性と信頼性を向上するために、各コントローラは複数のブ

10

20

30

40

50

ロセッサで構成され、前述の実行分担表はさらに各プロセッサの負荷状況とどのプロセッサが余裕があり負荷をバックアップしやすいかを示すものとし、これらの実行分担表とタスク負荷表に応じて故障したプロセッサの負荷をバックアップするプロセッサを定める内部バックアップ手段を設けたものである。

【0029】

【作用】一種類で多種多様な制御機能を並列に実行可能なコントローラを実現したことにより柔軟なシステム構成能力を有する分散制御システムが構築できる。即ち、ハード的に同一構造のコントローラを複数台設置することができるため一台のコントローラが故障しても他のコントローラでのバックアップが容易にでき、迅速な保守対応が困難な場所にもコントローラを分散設置できる。

【0030】プログラミングに関しては、ユーザは付与モデルとして比較的簡単なモデルをプログラムすればよい。学習の初期では付与モデルに従って動作をし、学習が進んでニューラルネットによる学習モデルの出力値が良くなるにつれてニューラルネットの出力に対する重み値を大きくしていくことができる。そしてさらに、状態の評価部を設け、その出力の時間微分によって出力データの評価を行うことで、前記重み値をユーザが意識することなく、学習によって次第に変化させていくことができる。また、該重み値を入力データからニューラルネットで求め、さらにそのニューラルネットを学習させることによって、付与モデルの出力が良いところではその出力を、学習モデルの出力の方が良い場合にはその出力の重みが大きくなるように場合によってその重み付けの仕方を変えることもできるようになる。

【0031】また、評価値の表示手段と修正手段によって、評価値とユーザの感覚とにずれがあった時に修正することで、簡単にユーザの感覚にあった評価が可能となる。また、ユーザが新しい評価基準を設けることで、さらにユーザの評価方法にできるだけ近付ける。そして、こうしてできた複数の評価基準を直接または何らかの変換をしてその最小値をとることによって、現在ユーザが最も不適と感じている項目を抽出し、この評価基準を用いて制御出力の演算法を更新していくことで、複数の評価基準からみて調和のとれた制御が可能になる。

【0032】故障診断時のネットワーク負荷を低減する手段に関しては、各コントローラは、故障発生時に異常情報を入出力フィルタに送り、入出力フィルタは、送信された異常情報を含んだ異常情報がすでにネットワークに送信されているかどうかを各コントローラが格納している履歴情報を参照して判定し、送信された故障情報を含んだ異常情報がすでにネットワークに送信されていないければ該異常情報をネットワークに送信することにより、冗長な異常情報がネットワークに送信されることを防止することができる。

【0033】さらに自律的なバックアップに関しては、

各コントローラに具備した実行分担表と、タスク負荷表と、受付順位表とにより、どのコントローラのバックアップ依頼手段でバックアップすべきタスクとバックアップに適するコントローラを検索し、バックアップ受付手段に通知することが可能になる。また、バックアップを依頼されたコントローラのバックアップ受付手段は自己が持つ実行分担表の自己の負荷に基づき、この通知にあるバックアップ対象タスクを実行可能か否かを自ら判断し、選択的にバックアップ対象タスクを受付ける。この選択の結果を受付メッセージとして放送（ブロードキャスト）することにより依頼元および他のコントローラはバックアップを実施したコントローラの負荷状況の変化を実行分担表及び受付順位表に反映可能である。さらに依頼元のコントローラでは受付られなかったバックアップ対象タスクを知ることが可能であり、再び受付順位表（この時は先にバックアップを実施したコントローラは負荷率が高まるため受付順位表の先頭にいないであろう）にしたがって新たなコントローラに未受付分を依頼可能である。

【0034】このようにして本発明では各コントローラがシステムの実行状況を常に正確に把握し、また、バックアップを依頼されたコントローラが依頼を拒否可能であるため、自律的にバックアップすべきタスクを分割／分散して処理することができる。さらに、一つのコントローラに含まれる複数のプロセッサの一つが故障したときにおいても、内部バックアップ手段が実行分担表により負荷が小さいプロセッサを選択し、そのプロセッサに故障したプロセッサの負荷をすべてバックアップできるかを判定し、そのプロセッサが負荷オーバにならないようにバックアップを実施させ、残りの負荷がある場合にはさらに他のプロセッサを選択し同様にバックアップさせる。このように本内部バックアップ手段は各プロセッサが負荷オーバにならない様に故障したプロセッサの負荷を各プロセッサの負荷限界に収まるように分割し、複数のプロセッサで次々とバックアップさせていくものである。

【0035】

【実施例】以下本発明の実施例を図面により詳細に説明する。

【0036】図1は本発明の一実施例であるプラント制御システムを示す。制御対象プラント100は流量計等のアナログ形センサ111, 115, 119, 制御弁等のアナログ形アクチュエータ112, 116, 120, リミットスイッチ等のデジタル形センサ113, 117, 121および電磁弁等のデジタル形アクチュエータ114, 118, 122から構成されている。本例ではセンサ111とアクチュエータ112が一組、センサ113とアクチュエータ114が一組となり各々制御対象機器を構成している。本図では各制御対象機器には一つのセンサと一つのアクチュエータしか図示していない

が、これらは複数個ある場合もある。コントローラ 1, 2, 3, 4, 5, 6 は各制御対象機器に対応して配置されそれらはネットワーク 1000 で結合されている。

【0037】マンマシン装置 1200 はプラントの運転監視を行い場合によっては人手での運転指令を各コントローラに発する装置である。

【0038】次に、本発明の一実施例の具体例の一つである鉄鋼圧延制御システムについて説明する。参考のため、従来の鉄鋼圧延制御システムを図 2 により説明する。ペイオフリール 151, 152 は被圧延材のコイルであり、被圧延材 180 は入側処理ライン 153, ルーパ 154 を経由して圧延機 155, 156, 157, 158 にて圧延され最終製品 181 となりテンションリール 159 に巻き取られる。上記リール、圧延機等はモータ 140, 141, 142, 143, 144, 145, 146 で駆動される。各々のモータに対応したドライブ装置 130, 131, 132, 133, 134, 135, 136 は各モータを駆動するパワー回路である。従来はドライブ装置 130, 131, 132, 133, 134, 135, 136 は電力変換器 190 とそれを制御する専用の高速コントローラ 195 から構成されている。図 2 にはドライブ装置 130 の内部のみ図示したがドライブ装置 131, 132, 133, 134, 135, 136 の内部構成も同様である。一方コントローラ 1500, 1501, 1502, 1503 は板厚目標値、入側板厚、スタンド間張力などから各モータのスピード指令を計算して各ドライブ装置に指令値を出力する構成を取っていた。

【0039】本制御システムの運転監視はマンマシン装置 1200, 1201 にて行う。各コントローラ及びマンマシン装置の間はネットワーク 1000 で結合されている。

【0040】図 3 は本発明を適用した鉄鋼圧延制御システムを示す。本実施例では各モータドライブ装置 161, 162, 163, 164, 165, 166, 167 に一対一でコントローラ 11, 12, 13, 14, 15, 16, 17 を割り付けている。しかも、各モータドライブ装置 161, 162, 163, 164, 165, 166, 167 は電力変換器 190 のみから構成されており、従来のような専用のコントローラはない。後述するように本発明のコントローラはマルチプロセッサ構成であり、従来の電力変換器制御機能はそのコントローラ内の一つのプロセッサに割り当てている。

【0041】なお、図示はしていないが圧延機 155, 156, 157, 158 のロール間ギャップの制御は通常、油圧圧下装置で制御されている。この油圧圧下装置はやはり数ミリ秒という高速で制御しなければならず、従来はそれ専用のコントローラを特別に配置していた。しかし、本発明によれば、モータの制御と同じくその制御に圧下指令計算用コントローラ内の一つのプロセッサ

を割り当てればよい。このような構成を採ることにより後述するように、非常に高信頼、高性能な制御システムが実現できる。

【0042】図 4 及び図 5 は本発明の他の一実施例であるところの自動車の制御を示したものである。図 4 は一般的な自動車の制御項目を説明したものである。図 5 は自動車の制御に本発明を適用した場合の制御システム構成を示したものである。

【0043】コントローラ 21 はエンジン 21a を制御する。このコントローラ 21 はシリンダに流入する空気量と、エンジンの回転数と回転角度と、アクセルペダルが踏み込まれた角度と、エンジンの冷却水の温度と、排気浄化用の触媒の温度とを検出する。これらに応じて、シリンダ内の燃料と空気の比が理論空燃比となるように燃料の噴射量を操作する。そして、上記の状態量に応じて点火時期を操作する。これらの操作はエンジンの効率を高め出力を増加する。同時に、排気中の有害物質を低減する。また、このコントローラ 21 はエンジンの冷却水の温度と排気浄化用の触媒の温度が適温より低い場合にはそれらを素早く暖めるために点火時期をやや進角する。コントローラ 21 は前照燈の点灯と冷房装置の運転をコントローラ 28 がネットワーク 1000 を介して通知したとき、エンジンの停止や車速の変動を防ぐためにスロットルをやや開いてエンジンの出力を増加させる。コントローラ 22 がネットワーク 1000 を介して通知する車速が特定の値を超過した場合には、コントローラ 21 は燃料の噴射量を減じて車速が特定の値を大きく越えないようにする。コントローラ 22 が変速中であるとネットワーク 1000 を介して通知したとき、このコントローラ 21 は変速の前後で加速度が滑らかに変化するようにエンジンの出力を調整する。コントローラ 21 はアクセルペダル踏み込み角度が急激に大きくなった場合には運転車が急激な加速を要求していると判断する。そして、このコントローラ 21 は冷房装置で消費されているエンジンの出力を車の加速に使用するためにネットワーク 1000 を介してコントローラ 28 に対して冷房装置の停止を要求する。同時に、コントローラ 22 にシフトダウンを要求する。

【0044】コントローラ 22 は変速機 22a を制御する。このコントローラ 22 は車速を検出し、これに応じたギヤ比に変速する。この時、ネットワーク 1000 を介してコントローラ 21 が通知するスロットル開度とエンジンの回転速度とコントローラ 23 が通知する車体姿勢に応じて、変速の基準となる車速を変更する。スロットル開度が小さくエンジンの回転速度が高いときはエンジンの吸気抵抗による出力の損失が増大するため、低速でシフトアップしエンジン回転速度を低下させる。車体姿勢が後傾しているときは坂道を昇っていると判定しシフトアップの基準車速を高め、低速ギヤを使用するようにして登坂力を維持する。

【0045】コントローラ23はサスペンション23aを制御する。このコントローラ23はサスペンションの変位と車体姿勢とを検出し、ネットワーク1000を介してコントローラ22から車速を、コントローラ25から舵角を受信し、これらに応じてサスペンションの空気バネ内の気圧を制御して所望の車体姿勢にする。このコントローラ23は車体姿勢をネットワーク1000を介して他のコントローラに通知する。

【0046】コントローラ24はブレーキ24aを制御する。このコントローラ24はタイヤの回転速度を検出し、ネットワーク1000を介してコントローラ22から車速をコントローラ25から舵角を受信し、これらに応じて地面に対するタイヤの滑り率を求め、この値が所定の値以下になるように制動力を定める。さらに、旋回時に内輪に弱い制動を掛けて滑らかに旋回する。

【0047】コントローラ25はパワーステアリング25aを制御する。このコントローラ25はステアリングシャフトのねじれ角度を検出し、その角度が小さくなるようにパワーステアリングのアシストトルクを調節する。さらに、このコントローラ25は舵角を検出し、他のコントローラにネットワーク1000を介して通知する。

【0048】コントローラ26は故障を診断する。このコントローラ26はネットワーク1000を介して他のコントローラから通知される状態量や操作量を自己のシミュレーション結果と比較することによって、他のコントローラあるいは制御対象機器の故障あるいは劣化を検出する。故障あるいは劣化を検出した場合は、コントローラ28に対して故障診断結果を通知する。

【0049】コントローラ27はドライブレコーダであり、ネットワーク1000を介して通知された車の動作の履歴を衝撃及び熱に強い記憶装置に格納する。

【0050】コントローラ28はスイッチ28aとメータ28bとディスプレイ28cなどのマンマシンインタフェースを制御する。コントローラ28は運転者がスイッチ28aを操作した場合にそのオン/オフに応じて対応する装置を作動あるいは停止させる。また、ネットワーク1000を介して通知された車速やエンジンの回転速度等の状態量と故障診断の結果などをメータ28bあるいはディスプレイ28cに表示する。また、ネットワーク1000を介した他のコントローラからの依頼あるいは要求に応じて各種装置をオン/オフする。

【0051】このように各コントローラはネットワーク1000を介して検出した状態量と制御出力あるいは操作依頼を互いに通知し、そして、各コントローラは通知された状態量と制御出力と操作依頼に応じて所轄の機器を制御する。

【0052】以上のように複数のコントローラを各制御対象機器毎に対応して分散配置するところに本発明の一つの特徴がある。

【0053】図6は本発明を階層システムに適用した場合のシステム構成例である。プラントが各々異なった制御対象機器群100、101、102から構成されている場合やその制御機器群100、101、102が広域に分散している場合などでは本図のようにネットワークをネットワーク1000、1001、1002のように分割し各ネットワーク下にコントローラ1、2、3、4、5、6を配置する。ネットワーク1000、1001、1002は各々統括コントローラ1300、1301、1302を介して上位ネットワーク1500に接続される。統括マンマシン1600はこの制御システムの運転監視を行う。

【0054】図7及び図8は今まで述べたコントローラの内部構成を示す。コントローラ1はグローバルメモリ(GM)52、演算プロセッサ(PR1、PR2、PR3)54、56、58入出力処理装置(I/O)53、ネットワーク制御プロセッサ(NCP)40、ネットワークコネクタ(NetworkConnector)41、規模拡張用コネクタ(BusConnector)60、61、62、63および制御対象機器との接続用コネクタ(I/O Connector)64から構成されている。また演算プロセッサ(PR1、PR2、PR3)54、56、58は各々ローカルメモリ(LM1、LM2、LM3)55、57、59を有している。

【0055】NCP40はバス70を介してGM52、LM55、57、59およびI/O53の内容を読み込み他のコントローラまたはマンマシン装置(図1の1200)へ送信する機能と、逆に他のコントローラまたはマンマシン装置2000から送信されたデータをGM52、LM55、57、59およびI/O53に書き込む機能を有する。バス72はプロセッサ54、56、58がGM52およびI/O53をアクセスするのに用いる。バス71は伝送用のシリアルデータ用のバスである。このような構成を採ることでコントローラの拡張性を確保でき、もし一つのプロセッサが故障しても他のプロセッサで代替処理ができ(後述)しかも1台のコントローラ内の全てのプロセッサが故障したとしても他のコントローラで代替処理ができる(後述)システムが構築できる。

【0056】図8は図7におけるプロセッサ(PR1)54の内部構成を示したものである。図7において各プロセッサは同一の構成であり、その構成をプロセッサ(PR1)を例に取り上げ説明する。プロセッサ(PR1)54はタスクを実行する汎用エンジン(GE)6811と、タスクの中に含まれたニューロ演算を加速するためのニューラルエンジン(NE)6812と、タスクの中に含まれたシーケンス制御用の演算を加速するシーケンスエンジン(SE)6813と、アクセスすべきメモリがグローバルメモリ(GM)52のときバス70とプロセッサ内のバス6819を結合するバスインタフェース

(BIF) 6817と、ローカルメモリ (LM) 55とグローバルメモリ (GM) 52の間で高速な転送を実行するダイレクトメモリアクセス制御 (DMAC) 6818と、スケジューリングのための時計であるフリーランタイマ (FRT) 6815と、各構成要素の故障を検出してプロセッサの動作を停止させるとともに故障を他のプロセッサにバス72を通して通知する故障制御回路6820を含む。さらに、各構成要素はバス6819で接続されている。

【0057】図9は今まで述べたコントローラの他の実施例の内部構成を示す。図7との相異点は図7におけるNCP40の機能を複数のプロセッサ84、87、90、93いずれにおいても実行できるように各プロセッサ84、87、90、93にシリアルデータの平行データ化あるいはその逆の操作と符号化／復号化などを実行するランアダプタ (LA) 86、89、92、95を付加したことである。これにより、プロセッサの代替処理と同様に伝送機能 (NCP) の代替処理が行える。

【0058】図10は図9におけるプロセッサ (PR1) 84の内部構成を示したものである。図8との相異点はランアダプタ (LA) 86とバス6819の接続インタフェース6902を設けたことである。

【0059】図11はさらに他の実施例の内部構成を示す。図9との相異点は図9におけるローカルメモリ85、88、91、94の代替としてキャッシュメモリ (CM) 301、302、303、304を用いたことである。キャッシュメモリコントローラ (CMC) 401、402、403、404はキャッシュメモリ (CM) 301、302、303、304とグローバルメモリ400との間のデータ転送を行う。

【0060】図12は図11におけるプロセッサ (PR1) 84の内部構成を示したものである。図10との相異点は図12においては図10におけるダイレクトメモリアクセス制御 (DMAC) 6818が不要なことである。これは、キャッシュメモリ (CM) 301とグローバルメモリ400との間のデータ転送をキャッシュメモリコントローラ (CMC) 401が行うためである。

【0061】図13はさらに他の実施例の内部構成を示す。図11との違いはランアダプタ86、89、92、95とネットワークコネクタ41およびプロセッサ84、87、90、93の接続をプログラマブルスイッチアレイ (PSW1、PSW2) 8101、8102でプログラマブルにしたことにある。これにより伝送機能をプロセッサ84、87、90、93いずれにおいても実行できるようだけでなく、シリアルデータの平行データ化あるいはその逆の操作と符号化／復号化などを実行するランアダプタ (LA) 86、89、92、95の故障に関してもランアダプタを切り替えることによりプロセッサを切り替えることなく伝送機能の代替処理が行える (図11の構成では、ランアダプタの故障であっても

プロセッサを切り替える必要がある。) 。尚、図13におけるプロセッサの内部構成は図12に示したものと同様で良い。

【0062】図14にはプログラマブルスイッチアレイの構成例を示す。スイッチアレイへの入出力 X_i と Y_j の交点にはMOSトランジスタ T_{ij} が設けられており、MOSトランジスタ T_{ij} のゲートには不揮発性メモリセル M_{ij} が接続されている。すなわちメモリセル M_{ij} の出力が“1”の場合MOSトランジスタ T_{ij} がONし X_i と Y_j が接続され事になる。逆に M_{ij} の出力が“0”の場合は T_{ij} がOFFとなり X_i と Y_j は切り離された状態となる。またメモリセル M_{ij} にはアドレス線 A_1 、 A_2 、 A_3 とデータ線 D_1 、 D_2 、 D_3 が接続されており、これを介してデータの書込みが行われる。

【0063】図15にメモリセル M_{ij} の内部回路例を示す。この回路では不揮発性メモリ素子としてフラッシュメモリを用いている。通常動作状態では書換えモード信号は“LOW”になっており、NMOS : N_1 がOFF、PMOS : P_1 がONとなっている。またアドレス線の電位は例えば電源電圧に等しくなっている。従ってノードAの電位はメモリ素子が消去状態 (しきい値電圧が低い) のときは“LOW”となりインバータI2の出力も“0”となる。すなわちこのセルにつながるMOSトランジスタ T_{ij} がOFFとなる。逆にメモリ素子が書込み状態 (しきい値電圧が高い) のときはノードAの電位が“HIGH”となりインバータI2の出力が“1”スイッチMOSトランジスタ T_{ij} がONとなる。メモリセルへの書込みを行うには書込みモード信号を“HIGH”にし、NMOS : N_1 をON、PMOS : P_1 をOFFにする。この状態でアドレス線及びデータ線に高電圧を印加することによりメモリ素子 M_1 は書込み状態になる。メモリ素子の消去はアドレス線を“LOW”にしメモリ素子のソースに高電圧を印加することにより行われる。

【0064】図16にプログラマブルスイッチアレイに予備領域を設け、部分的な故障に対して救済を可能とするための構成を示す。この例は図14に示したスイッチアレイに故障救済用素子 : $T_{s1} \sim T_{s3}$ 、 $T_{1s} \sim T_{3s}$ 、 $T_{y1} \sim T_{y3}$ 、 $M_{s1} \sim M_{s3}$ 、 $M_{1s} \sim M_{3s}$ を追加したものである。通常故障のない状態では $T_{s1} \sim T_{s3}$ および $T_{1s} \sim T_{3s}$ はOFF、 $T_{y1} \sim T_{y3}$ はONとなっており、図14の回路と同じ動作をする。次に故障が検出された場合の救済方法を以下に示す。メモリセル M_{11} が故障した場合について説明する。まず $M_{11} \sim M_{13}$ の内容を $M_{s1} \sim M_{s3}$ に移す。次に M_{1s} を“1”にし、 T_{1s} をON、 T_{y1} をOFFにする。これにより故障部分が予備回路で置き換えられたことになり、故障前と同一の動作が可能となる。

【0065】図17は図8、図10、図12における汎

用エンジンの内部構成例を示したものである。本エンジンはマイクロプログラムメモリ8202、読みだし回路8203、マイクロ命令レジスタ8204及びデコーダ8205、レジスタ、演算ユニット8206、マルチプレクサ8208、アドレスデコーダ8207から構成される。起動当初はアドレスデコーダ8207がクリアされ、マイクロプログラムメモリ8202のゼロ番地の命令がデコーダ8205で解釈され実行される。マイクロ命令には次に実行する命令の番地を指し示している(8207)。そのようにしてマイクロプログラムメモリ8202に書き込まれているマイクロ命令が順次実行される。

【0066】マイクロプログラムメモリ8202は例えばフラッシュメモリ等の不揮発性メモリで構成されており、このマイクロプログラムを書換えることによりプロセッサの機能を目的にあったものに変更することができる。この書換えのためにマイクロプログラムメモリ8202は専用の書換え回路8201を内蔵しており、書換えモード信号8302により書換えモードに設定することにより書換え用のアドレスバス8301、データバス8300から供給されるアドレス及びデータにしたがって書換えが行われる。

【0067】図18に他の実施例として各プロセッサを複数のプロセッサエレメント(PE)の組み合わせで構成した例を示す。各プロセッサの目的に応じて必要な数のプロセッサエレメントを組み合わせ各プロセッサを構成する。こうする事によりLSI全体を同一のプロセッサエレメントで構成することができ、LSI化が容易になると共に無駄の少ないコントローラの構成が可能となる。

【0068】次に入出力部について詳細に説明する。

【0069】先にも述べたように各コントローラに接続されるセンサ、アクチュエータの種類及び点数は制御対象により異なる。従って従来はそれらに応じた入出力装置をレポートリの中から選択しコントローラに接続する必要があった。一方、本発明では極力少ない種類のハードウェアを用いて制御システムを構築するのが一つの目的であるから以下に示す入出力構成を採用した。

【0070】図19は図9におけるI/O83の内部構成を示す。I/O83はバスインタフェース830、D/A変換器832、OPアンプ833からなるアナログ出力とOPアンプ835、A/D変換器834からなるアナログ入力とレジスタ836、パワー回路837からなるデジタル出力とパワー回路838からなるデジタル入力及びこれらとI/Oコネクタ64との結合を決定するパワースイッチアレイ840から構成される。

【0071】パワースイッチアレイ840のプログラムはインタフェース回路830から信号線850を介して行われる。パワースイッチアレイ840の構成は図16で示したものと同一であるが、パワーを取り扱うため物理的な容量は大きくする必要がある。

【0072】図20はI/O83を1チップの半導体で実現した構成図を示す。半導体は大きく2層より構成され上部半導体980にはパワー回路部902、パワースイッチアレイ部901、プロセスとの接続端子900よりなり下部半導体950はOPアンプ部951、D/A変換部952、A/D変換部953、レジスタ部954、バスインタフェース955及びバスインタフェースとの接続端子960から構成されている(図21は下部半導体950を図示したものである)。また、上部半導体980と下部半導体950の配線は配線層970によって行う。

【0073】図22及び図23は図9におけるコントローラ1'を1チップの半導体で実現した構成図を示す。半導体は大きく2層より構成され上部半導体980は図20と同じ構成であるが、下部半導体950はOPアンプ部951、D/A変換部952、A/D変換部953、レジスタ部954以外にメモリ群956およびプロセッサ群957から構成されている(図23は下部半導体950を図示したものである)。また、上部半導体980と下部半導体950の配線は配線層970によって行う。

【0074】以上コントローラの内部構成について実施例を用いて説明したが、本発明の範囲は実施例の範囲に限定されるものではなく種々の変形が可能である。例えば機能プログラム用の不揮発性メモリとしては、フラッシュメモリ以外に他の電氣的に書換え可能なEEPROMまたは強誘電体RAM、さらには電池でバックアップしたSRAMでもよい。

【0075】次に図24に本発明のコントローラの外観を示す。コントローラの前面にはI/Oコネクタ64が左右側面と上下面には規模拡張用コネクタ60、61、62、63が実装されている。背面1700にはネットワーク用コネクタ1701及び電源用コネクタ1702が実装されている。外形寸法は高さ約20cm、奥行き約10cm、幅約3cmあり、入出力を接続可能でかつ分散設置するに十分な寸法である。本寸法は本発明によりどの程度の小型化を図れるかを参考にするためのものであり、本発明はこの寸法に限られる訳ではない。

【0076】図25は二台のコントローラを規模拡張用コネクタ60、61を用いて結合したものである。

【0077】図26は二台のコントローラを規模拡張用コネクタ62、63を用いて結合したものである。

【0078】図27は四台のコントローラを規模拡張用コネクタ60、61、62、63を用いて結合したものである。

【0079】このように本実施例のバス構成と外観構造を有することによりコントローラ規模が容易に拡張できる。

【0080】次に、本発明で用いた制御演算方法について説明する。

10

20

30

40

50

【0081】図28はセンサー等からの入力データに基づいて制御演算を行う制御演算部の基本構成図を示す。本発明では、入力データ2001を複数の出力生成手段、ここではニューラルネットワーク2101とファジィ推論手段2102とPID制御手段2103の3個の手段に入力する。そして、重み付け部2021において各出力生成手段の出力値をそれぞれに対応する重み値 K_1 、 K_2 、 K_3 で重み付けし、出力データ合成部2022で重み付けした各出力値を加算し、出力データ2002を得る。この時、前記重み値 K_1 、 K_2 、 K_3 を加えて1になるようにしておけば、出力データ合成部2022では、重み付けされた出力を単純に加算するだけでよい。ここで、ニューラルネットワーク2101、ファジィ推論手段2102、PID制御手段2103の3個の手段は各々並列的に実行できるため各々別プロセッサで実行させる（例えば、図7のコントローラでいえば、プロセッサ54、56、58に3種の演算を担当させる）ことにより、処理速度を落とすことなく制御演算が実行できる。

【0082】図29に実施例の全体概要図を示す。ここでは、出力計算部2012及び出力評価部2015よりなり、出力計算部2012および評価部2015がそれぞれ学習モデルと付与モデルを持って両者を適応的に協調させる制御方法を示す。センサ等2010からのデータ2001を必要に応じて入力処理部2011で入力処理した入力データ2300は出力計算部2012と評価部2015へ入力される。出力計算部2012では、入力データ2300を受けとり、その内部に持つパラメータ等を用いてその出力データ2301を計算し、必要に応じて出力処理部2013にて必要な出力処理を行った後、その値2002をアクチュエータ等2014へ出力する。ここで、パラメータ等とは、学習モデルであるニューラルネット内のニューロン間の結合の重み値や、付与モデル内のパラメータ、さらに学習モデルと付与モデルの重み付けのパラメータを含む。一方、評価部2015では、やはり、入力データ2300及び内部のパラメータ等を用いて現在の状態の評価を行い、その結果から出力評価信号2302を出力する。そして、出力計算部2012ではこの出力評価信号2302を用いて、この値が大きくなるように内部のパラメータ等の学習を行う。一方、評価部2015内のパラメータに関しては、例えばマンマシン装置1200に状態評価信号2303を表示し、ユーザがこれを見て不適切と感じる場合にタッチパネル、マウス等のポインティングデバイス2017を用いて修正値を入力する。これを外部評価信号2304として評価部2015へ入力し、この時に外部評価有効信号2305をオンにして評価部2015内のパラメータ等を更新することによって、様々な状況に適応した評価ができるように学習させていく。

【0083】出力計算部2012の構成を図30に示

す。出力計算部2012の学習は、評価部2015の場合の外部評価信号2304のような直接的な教師信号はないため、各パラメータ等に乱数成分を付加し、その時の評価値に基づいて乱数成分を評価し、修正を行う。従って、出力の計算法は以下になる。まず、入力データ2300と学習モデルパラメータ保持部2421から得られる学習モデルパラメータを用いて、学習モデル計算部2401において出力を計算すると共に、予めユーザから付与された付与モデルに従って、入力データ2300と付与モデルパラメータ保持部2411から得られる付与モデルパラメータに乱数付加部2412によって微小な乱数を付加したものをパラメータとして用い、付与モデル計算部2402の出力を求める。そして、重み付け部2403でそれぞれの出力値を重み付けして加えあわせて出力データ2301とする。ただし、学習モデル計算部2401からの出力には乱数付加部2422によって微小な乱数を加えたものに対し重み付けを行う。また、さらに、重み付け部2403で用いる重み値も、重み値保持部2431に保持されている重み値に対し、乱数付加部2432によって乱数を付加したものを実際に用いて重み付けする。まず、重み付け部2403の重み値に関しては、乱数付加部2432によって付加された乱数2433と評価部2015から得られる出力評価信号2302を重み値更新量計算部2434において、例えば乗算し、重み値保持部2431にその値を渡して、もとの重み値に加えて新しい重み値とする。一方、学習モデル計算部2401内のパラメータに関しては、出力誤差計算部2424にて、乱数付加部2422によって付加された乱数2423と評価部2015からの出力評価信号2302を例えば乗算して、学習モデル計算部2401に与える誤差とし、学習モデル計算部2401では、その値に基づいてバックプロパゲーション等の学習アルゴリズムに従って内部のニューラルネットの重み値等のパラメータに更新量を計算し、新たな値を学習モデルパラメータ保持部2421に書き込む。また、付与モデル計算部2402では、やはり、付与モデル内パラメータ更新量計算部2414において、乱数付加部2412で加えた乱数2413と評価部2015からの出力評価信号2302を例えば乗算し、その値を付与モデルパラメータ保持部2411に渡して新しい付与モデルパラメータの値を書き込む。

【0084】一方、図31に評価部2015の構成を示す。ここでは、まず、入力データ2300と学習モデルパラメータ保持部2521から得られるニューラルネットの結合の重み値などの学習モデルパラメータを用いて、学習モデル計算部2501から出力を求める。また、それと並行して、予めユーザから付与されたモデルに従って、入力データ2300と付与モデルパラメータ保持部2511から得られる付与モデルパラメータを用いて、付与モデル計算部2502においても出力を計算する。

そして、重み付け部 2503 でそれぞれの出力値を重み値保持部 2531 に保持されている重み値に基づいて重み付けして足しあわせ、その足しあわせた結果を時間微分計算部 2505 にて時間微分することによって出力評価信号 2302 を生成する。また、評価部 2015 内の学習に関しては、外部評価有効信号 2305 がオンの時に外部評価信号 2304 からの信号に基づいて行われる。例えば、学習モデル計算部 2501 と付与モデル計算部 2502 のそれぞれの出力を比較し、外部評価信号 2304 に近い方の重み付けパラメータを大きくし、もう一方のパラメータを小さくする。また、学習モデル計算部 2501 に対しては、外部評価信号 2304 を教師信号として与えて学習を行う。一方、付与モデル計算部 2502 では、通常の出力値以外に、パラメータに微小な乱数を与えてもう一つの出力値を求め、その両者を比較して乱数を加えた方が外部評価信号 2304 に近ければ、実際にパラメータに対し、加えた乱数と学習定数を掛けたものを足し、乱数を加えない方が良い場合には、パラメータから加えた乱数と学習定数を加えたものを引くことによって学習を行うことができる。また、付与モデルは学習機能がなくても良いし、評価部 2015 が付与モデル計算部だけで学習モデル計算部がなくても差し支えない。また、学習モデル計算部 2501 と付与モデル計算部 2502 の両方を有する場合には、初期状態の両者の出力の重み付けは、付与モデル側を 1 とし学習モデル側を 0 とすることによって、学習開始前には学習モデルの出力が影響を及ぼさないようにすることができる。

【0085】次に、複数の評価基準を有する場合の実施例を示す。全体構成図を図 32 に示す。ここでは、複数の評価部として第 1 評価部 2601 と第 2 評価部 2602 があるものとする。各評価部内のパラメータの修正は、評価部が 1 個の場合と同じく、マンマシン装置 1200 に第 1 評価部 2601 の状態評価値 2603 と第 2 評価部 2602 の状態評価値 2604 を表示し、これをタッチパネル、マウス等のポインティングデバイス 2017 を用いて修正し、修正された時は、該当する外部評価有効信号 2605 に 1 を立てる。そして、その時の修正値を外部評価信号 2606 として各評価部へ受け渡す。そして、該当する評価部では、その内部のパラメータを修正すれば良い。また、出力評価信号に関しても複数得られるが、例えば、その複数の出力評価信号の最小値を選んで出力計算部 2012 へ与えることが考えられる。すると、ユーザが最も不適であると感じる評価基準に関して、より良くなるように出力計算部 2012 内のパラメータ等を更新できることになる。

【0086】次に、複数の手段や方法による出力の重み付けをする重み値を、入力データ 2001 によって変化させる場合を図 33 に示す。ここでは、入力データ 2001 をニューラルネットワーク 2701 に入力し、出てきた出力に乱数付加部 2702 によって乱数成分を付加し、

その値 K に基づいて重み付け部 2021 において、ニューラルネットワーク 2101 と出力生成手段 2102 の出力を重み付けし出力データ合成部 2022 で重み付けされたデータを加えあわせて出力データ 2002 を生成する。一方、教師信号生成部 2703 においてその乱数成分と出力評価信号 2302 を掛け算しニューラルネットワーク 2701 の出力と足して教師信号としてニューラルネットワーク 2701 に渡し、学習を行わせる。ただし、この時、学習が始まる前はニューラルネットワーク 2701 の出力が 0 に近い値になるようにし、出力生成手段 2 (2102) の出力が出力データ 2002 になるようにすると良い。

【0087】以下、本発明のネットワーク通信負荷を低減する方法に関する実施例を図面を用いて詳しく説明する。図 34 は、本発明の対象となる分散制御システムの構成を示したもので、ネットワーク 1000 によって複数のコントローラ (1, 2, 3, ...) が制御を行っている。システム全体の制御情報および診断情報は、マンマシン装置 1200 において加工・表示される。各コントローラは、それぞれに制御を実行するだけでなく、診断処理も実行するので、診断処理部 4010、制御実行部 4011、制御情報および診断情報をネットワークを通じて他のコントローラに送信するための通信インターフェース 4012 をもっている。

【0088】このような分散制御システムに於ては、コモンモード故障が発生した場合に、各分散制御センサーからの異常情報が多数ネットワークに送信され、通信不良となる可能性がある。

【0089】例えば、各コントローラが制約条件をもっており、診断処理部において制約のチェックをおこない、もし、制約が充足されなければ制約を充足しない制約式をネットワークに送信する場合を考える。このような場合、異常診断とは充足されない制約式の制約変数の中でどの変数が制約を充足しない原因となっているかを同定することである。

【0090】図 35 は、以後の説明のための具体例として 4 つのコントローラからなる分散制御システムの各コントローラの入力変数、出力変数、制約式を示したものである。例えば、コントローラ 1 は、入力変数として X_1 、 X_2 をもち、出力変数として X_3 をもち、制約式として、 $C1: E(X_3, X_1 + X_2, e_1)$ をもつとする。ここで、 E は、誤差関係式で、第 1 変数と、第 2 変数の絶対値の差が第 3 変数以下であることを表す。この例では、

$$X_1 + X_2 - e_1 < X_3 < X_1 + X_2 + e_1$$

であることを示している。

【0091】対象プラントの状態量が $X_1 = 1$ 、 $X_2 = 2$ 、 $X_4 = 2$ とする。コントローラが、この各状態量に対して制御を行う時、図 35 に示した制約をおのおのがチェックして、もし制約が充足されない場合には、ネッ

トワークに情報を送信するとしよう。この時、コントローラ 1 が正常に動作していれば、入力変数 X_1 , X_2 の値から、制約条件 C_1 を充足する X_3 の値である $X_3 = 3$ を決定し、出力する。しかし、ここで、コントローラ 1 が何等かの原因で故障し、 $X_3 = 13$ を出力したとしよう。この時、コントローラ 2 では、制約条件 C_2 と、入力 X_3 , X_4 から、 $X_5 = 15$, 制約条件 C_3 と入力 X_3 から、 $X_6 = 13$ が出力されるが、制約条件 C_4 が充足されないので制約条件 C_4 が未充足であるという異常が送信される。コントローラ 3 では、入力 X_3 , X_5 と、制約条件 C_5 より、出力 $X_7 = 195$ となるが、これは、制約条件 C_6 を充足しないのでコントローラ 3 からは、制約条件 C_6 が未充足という異常が送信される。コントローラ 4 では、入力変数 X_3 , X_5 および X_6 と制約条件 C_7 より $X_9 = 15$ が出力され、制約条件 C_8 より、 $X_8 = 26$ が出力されるが、これは、制約条件 C_9 を充足しないので、コントローラ 4 からは、制約条件 C_9 が未充足という異常情報が送信される。このように、分散制御系においては一つのコントローラが故障したために、他の多くのコントローラから異常情報が出力され通信ネックが発生し、診断が妨げられる心配がある。

【0092】このような問題を解決するために、本発明においては、個々のコントローラからネットワークへ入出力される情報をフィルタリングする方法をとった。また、異常発生時に送信する情報として、各コントローラがフィルタリングを実行するためにマンマシン装置に送るものと、ネットワークを経由して順次各コントローラに送るものの二つデータフレームを用意する。ネットワークもこれにあわせて、各コントローラをリング状に結合するものと、各コントローラとマンマシン装置をつなぐ二種類のネットワークを用意すれば、より高速な処理が可能である。ここでは、一実施例として、マンマシン装置も各コントローラも一つのネットワークで接続しており、異常発生時の異常情報の送り先を異常情報の中に含める方法をとる。

【0093】図 36 は、個々のコントローラの通信インタフェースの構成を示したもので、4030 が不要な入力情報をふるい分け、必要な情報だけコントローラに取り込む為の入力情報フィルタ部、4031 は、不要な出力情報をふるい分け、不要な出力情報をネットワークに送信しないための出力情報フィルタ、4032 は、暗号化され、情報圧縮された情報を復元するための通信処理部である。これらの処理は図 7 で示すコントローラでは NCP40 が、図 9、図 11 に示すコントローラではランアダプタ 86, 89, 92, 95 が実行する。

【0094】ここで、不要な出力情報のふるい分けの一実施例として、制約変数集合を用いた方法を説明する。ここで、制約変数集合とは、制約式中に定義された全ての変数をさす。例えば、図 35 中の制約条件 C_1 の制約

変数集合を $V(C_1)$ とすると、 $V(C_1) = \{X_1, X_2, X_3\}$ である。誤差変数 e_1 は誤差範囲を示すために人工的に導入されたものなので、制約変数集合には含めないとする。

【0095】この制約変数集合の包含関係に基づいて、個々の分散制御系で検出した異常をネットワークを経由して、他の個別コントローラとマンマシン装置 1200 に送信するか否かを決定する。ただし、この情報だけでは不十分なので、個々の分散制御システムにおける入出力変数の情報も利用する。つまり、ネットワークに異常情報を送信するときは、図 37 に示したように、制約変数集合の他に、入力変数集合、展開変数集合、消去変数集合、異常情報データ ID、機器 ID の情報も付加する。ここで、入力変数とは、異常が発生した個々のコントローラにおける入力変数で、コントローラ 1 における入力変数を、 $I_n(1)$ とすると、 $I_n(1) = \{X_1, X_2\}$ である。展開変数と、消去変数は、別のコントローラから異常情報を受信した場合に異常変数の絞り込みの際につけ加わる情報で、後で詳しく述べる。異常データ ID は、このデータフレームが、異常データというタイプのデータで、いつ送信されたかを示す為の情報で、ここでは、AB_92_07_03_10_24_10 とする。AB は、異常情報であることを示し、後の数字は異常情報を送信したのが、92 年 7 月 3 日 10 時 24 分 10 秒であることを示す。送信先 ID は、異常データが、マンマシン装置あつてのものか、各コントローラ宛のものかを示すものである。例えば、マンマシン装置向けのものであれば、ここに、0 を、各コントローラ宛のものであれば、ここに、また、機器 ID は、この異常情報が発せられたコントローラの ID を記述するためのデータフィールドでここでは、コントローラ 1 から送信されたので 1 と記入されている。

【0096】このような異常情報がネットワークに送信されると、各コントローラは、その情報が自分と関係のあるものかどうかを、入力情報フィルタにおいて判定し、もし関係のある情報であれば取り込む。一方、異常情報データフォーマット中の異常データ ID と制約変数集合、機器 ID、入力変数集合だけは、加工されないまま、ただちに、マンマシン装置に送られる。

【0097】以下の説明では、今の例をもちいて、まず、どのようにして不必要な異常情報が、ネットワークに送信されるのが抑制されるかを示し、次に、異常箇所がどのように同定されるかを示す。

【0098】まず、不必要な情報の抑制処理を図 38 及び図 39 に示す。図 38 は、処理の概要を示したもので、マンマシン装置に登録されている異常情報を参照して、新たに送信しようとしている異常情報が既に送信されている情報に含まれているかどうかを判定し、もし含まれていれば、送信を止める。図 39 は、異常情報相互の包含関係の判定処理を示したもので、制約集合の間に

10

20

30

40

50

包含関係があるかないかをまず調べ、もし包含関係がなくてもコントローラ間に入出力関係があり、この入出力関係まで考慮すると包含関係が成り立つ場合もあり、この判定も行っている。これらの処理を具体的な例で示す。

【0099】まず、コントローラ1から上記の異常情報がマンマシン装置宛のものと各コントローラ宛のものの2つのデータフレームとして送信され、すでにマンマシン装置で受信している場合について考えてみる。すでに述べたように他のコントローラ2, 3, 4でも制約違反を検知し異常情報を送信しようとする。たとえば、コントローラ2では、制約条件C4が充足されない。この異常情報の制約変数集合は $V(C4) = \{X5\}$ であり、入力変数集合は、 $\{X3, X4\}$ である。コントローラ2の通信インタフェースの出力情報フィルタ部では、この情報を送信する前に、マンマシン装置の異常情報を参照して、自分が発信しようとしている異常情報の制約変数集合を含む制約変数集合をもった異常情報がすでに送信されていないかどうか確認する。この場合、コントローラ1から送信された異常情報の制約変数集合の $V(C1) = \{X1, X2, X3\}$ は、 $V(C4)$ を含んでいない。しかし、入出力関係まで考慮すると異なった結果となる。つまり、コントローラ2の入力変数X3は、コントローラ1から送信された異常情報の制約変数集合 $V(C1)$ に属しているから、この入力変数X3の影響でコントローラ2の出力変数X5, X6に異常が生じる可能性がある。

【0100】一方、コントローラ1の異常情報の入力変数集合は、 $\{X1, X2\}$ であるから、コントローラ2の出力変数X5に関する異常の原因は、コントローラ1の出力変数X3を媒介してコントローラ1の入力変数X1, X2である可能性がある。つまり、コントローラ2にとっては、 $V(C1) = \{X1, X2, X3\}$ にくわえて、 $Op(C1, 2) = \{X5, X6\}$ も制約変数である。ここで、 $Op(C1, 2)$ は、コントローラ2にとっての制約条件C1の仮想的な制約変数であって、ここでは展開変数集合と呼ぶ。そこで $V(C1) \cup Op(C1, 2)$ に $V(C4)$ が含まれているから、制約条件C4の未充足は、すでに送信された制約条件C1の未充足に関する情報に含まれる情報であって、ネットワーク上にデータがあふれることを防止するためには、出力情報フィルタ部でふるいにかけ、送信させないこととする。

【0101】同様にコントローラ3においても制約条件C6が充足されていないのでこれをネットワークに送信しようとする。ところがすでに、コントローラ1から異常情報がマンマシン装置に送信されているので、コントローラ3の出力情報フィルタでは、自分の持っている異常情報が送信するに値するものかどうか判定する。この時、コントローラ2におけるのと同様に、 $Op(C1, 3)$ を求めると、コントローラ3は、入力変数としてコ

ントローラ1の出力変数X3を持っているからコントローラ3の出力変数X7も仮想的な制約変数であって、 $Op(C1, 3) = \{X7\}$ となり、 $V(C1) \cup Op(C1, 3)$ に $V(C6)$ が含まれるから、制約条件C6に関する異常情報は、送信されない。さらに、コントローラ4についても $Op(C1, 4) = \{X8, X9\}$ であり、 $V(C1) \cup Op(C1, 4)$ に $V(C9)$ が含まれる。したがって制約条件C9が充足されないための異常情報は送信されない。このようにして、重複した異常情報の発生を抑えることができる。

【0102】次に、既に送信された異常情報を用いて、異常情報を絞り込んでいく方法について図40によって説明する。異常情報が、あるコントローラから送信されるとデータは、直接、マンマシン装置に行くものと、各コントローラに放送されるものに分かれる。異常情報の絞り込みが行われるのは、各コントローラに放送される異常情報である。まず、各コントローラはネットワーク上の情報が、自分と関係のある情報かどうかの判定をおこなう。

【0103】まずデータバス上の異常情報をコントローラの通信インタフェースに取り込み、取り込まれた異常情報は、ネットワークから削除する。通信インタフェースでは、まず通信処理部において、データを圧縮された状態から処理可能な状態に変換し、次に異常情報データの中の入力変数を出力変数として持つかどうかを判定し、もし持てば、コントローラの入力変数を展開変数として異常情報に付加する。今の例では、コントローラ1から発生した異常情報の入力変数集合は、X1, X2であり、これらをコントローラ2は、出力変数としてもたないからNoである。次に、異常情報の制約集合に含まれる制約変数集合に含まれる制約変数集合にもつ制約条件はあるかを判定し、これもNoであるから、異常情報を加工することなく、ネットワークに異常情報が返される。つまり、入力フィルタでフィルタされ、何も処理がおこなわれない。今の例では、他のコントローラも同様にして、何の処理も行わず、始めに送信された制約C1の異常情報が原形のまま、ネットワークをとおりぬける。一方、既に述べたようにこの場合には、ほかの制約違反は、出力情報のフィルタにかかって、発信されないから、絞り込みがなくても、はじめから原因は、コントローラ1にあると特定することができる。これに対して、何かの原因で、例えば、コントローラ3のC6の制約条件違反の異常情報が、コントローラ1からよりも先に送信されたとする。この異常情報の制約変数集合は、 $V(C6) = \{X7\}$ であり、入力変数集合 $In(3) = \{X3, X5\}$ である。この異常情報が、コントローラ1にくると、入力変数X3をコントローラ1は、出力変数としてもつから（このような変数を共通変数という）、コントローラの入力変数X1, X2, およびを展開変数に追加する。次の判定で、異常情報の制約

変数集合 {X7} と展開変数集合 {X1, X2, X3} の和集合は、制約条件1の制約変数集合 {X1, X2, X3} を含むから、異常情報は、図41の処理手順によって加工される。このとき、制約条件C1は、制約条件C6に含まれるという。まず、受信した制約条件に含まれる制約条件C1が充足しているかどうかを判定する。この場合は、充足していないので、次に、制約条件C1は、送信されているかどうかをチェックする。C1が先に送信された場合には、C6は、すでに述べたプロセスにより送信されないが、C6が先に送信された場合には、C1が送信されることはありえる。C1がすでに送信されていれば、受け取った異常情報は、消去され、まだ送信されていなければ、異常情報は、C1に関するものに書き換えられる。いずれの場合にもネットワーク上には、C1に関する制約条件のみが生き残る。これは、見方をかえれば、異常原因がコントローラ1に絞り込まれたことをしめしている。

【0104】なお、以上のような不必要な情報の送信のカットと、異常原因の絞り込みは、各コントローラの出力フィルタの部分及び、マンマシン装置に異常情報の包含関係に関する知識をインストールすることによっても実現できる。

【0105】例えば、制約の包含関係を図42に示したようなツリーで表現した知識を用意し、図37の異常データフォーマット上の制約変数集合のかわりに、充足されなかった制約名称を記述することとすれば、図39の処理の代りに既にマンマシン装置に登録されている異常情報の制約式名称を用いて、既に登録されている異常情報が送信しようとしている異常情報を含んでいれば、異常情報を他のコントローラに送信せずに、統合監視装置の異常情報を追加するだけにすることにより、不要な異常情報の送信を抑制できる。

【0106】さらにまた、信頼性及び耐故障性の向上を目的とした本発明の分散制御システムのもう一つの実施例を示す。本発明は分散制御システムのコントローラの多重ダウンに対しても稼働中のコントローラが負荷を自律的に分散しバックアップすることを可能にしたものである。このバックアップを実現する手段を図43に示す。図43は分散制御システムのコントローラの多重ダウンに対して稼働中のコントローラが負荷を自律的に分散しバックアップする本発明の一実施例である。プラント100のセンサ及び制御対象に接続されそれらを制御するコントローラ1, 2, 3, 4はネットワーク1000で相互に接続された分散制御システムを構成している。コントローラ1, 2, 3, 4はそれぞれ通信手段6110, 6210, 6310, 6410と、バックアップ依頼手段6120, 6220, 6320, 6420と、バックアップ受付手段6130, 6230, 6330, 6430と、スケジューラ6140, 6240, 6340, 6440と、制御手段6150, 6250, 6350, 6450と、記憶手段6160, 6260, 6360, 6460と、さらに、内部バックアップ手段6170, 6270, 6370, 6470と、切り離し手段6180, 6280, 6380, 6480とを含む各コントローラは同様の構成であるので、コントローラ1を代表例として取り上げ説明する。なお、各コントローラは後述の図47に示すとおり4個のプロセッサを持っている。さらに各プロセッサはここに示した通信手段、バックアップ依頼手段、バックアップ受付手段、内部バックアップ手段、切り離し手段、スケジューラ、制御手段の各手段を提供するに十分なハードウェアを持ち、かつ、各手段を提供するソフトウェア（タスク）を実行するに十分な処理能力を持っている。このため前記の手段は何れのプロセッサ上でも実施可能である。以下各手段の機能を説明する。

【0107】通信手段6110はネットワーク1000、バックアップ依頼手段6120、バックアップ受付手段6130、スケジューラ6140、制御手段6150と記憶手段6160、内部バックアップ手段6170、切り離し手段6180に接続されており、他のコントローラとのメッセージの交換を実行し、コントローラ内の他の手段と他のコントローラの間でのメッセージあるいはデータの交換を可能にする。スケジューラ6140は自分自身に登録された複数のタスクをプロセッサに割り付ける順番及び時間間隔などを定め、かつ指示するものである。制御手段6150はプラント100のセンサ及び制御対象を制御するものである。スケジューラ6140と制御手段6150は記憶手段6160と接続されており、そこに記憶されたタスク及びデータとスケジューリング情報に基づきスケジューリングとプランと1000の制御を司る。また、スケジューラ6140と制御手段6150は通信手段6110と接続されており、他のコントローラとの同期をとるために通信手段6110とネットワーク1000を介して他のコントローラとメッセージを交換する。本実施例は以上により分散制御を実施している。

【0108】バックアップを司る手段について説明する。

【0109】切り離し手段6180は内部バックアップ手段6170と通信手段6110に接続されている。この手段はコントローラ内部のハードウェアの故障あるいはソフトウェアの暴走などのダウンを検出し、ダウンしたプロセッサの動作を停止させる。また、ネットワーク1000を通して他のコントローラから自己を含むコントローラの故障を通知された場合にも同様の動作を実施する。他のコントローラからの故障の通知によってバックアップを開始できるように、切り離し手段6180は通信手段6110に接続されている。さらに、自ら故障を検出した場合および故障を通知された場合のどちらでも、どのプロセッサがダウンしたかを内部バックアップ

手段6170に前記の接続を通して知らせる。

【0110】前述のスケジューラ6140は、前述の機能に加えて、コントローラ内部のプロセッサで実行されている各タスクに対して単位時間当りの実行時間を測定し、その値の単位時間に対する比を各タスクの負荷率とする。この値により後述するタスク負荷表を更新する。そして、これらの負荷率の和が後述する限界負荷率を越えた場合に当該のプロセッサにて負荷オーバが発生すると予測する。また当該のプロセッサにて各タスクの負荷率の和が100%を越えたか否かを判定することにより負荷オーバを検出する。越えた場合が負荷オーバである。負荷オーバが発生すると予測されたプロセッサあるいは負荷オーバが発生したプロセッサを内部バックアップ手段6170に知らせる。

【0111】内部バックアップ手段6170は通信手段6110、記憶手段6160、切り離し手段6180、スケジューラ6140とバックアップ依頼手段6120に接続されている。この手段は、切り離し手段6180からコントローラ内部のダウンを通知された場合に、あるいはスケジューラ6140からコントローラ内部の負荷オーバの発生あるいは負荷オーバの予測を通知された場合に図44の手順に従い、ダウンあるいは負荷オーバにより実行できなくなるタスクをその優先度に応じてコントローラ内でバックアップする。部分的ダウンあるいは負荷オーバがコントローラ全体のダウンにつながることを防ぐために、コントローラの管理運用に必要なタスクを最も優先する。ネットワーク1000の通信量のボトルネックが顕在化し通信の遅延あるいは不良が頻発することを防ぐために、他のコントローラでバックアップした場合に多量の通信を必要とするタスクが次に優先される。バックアップは負荷の軽いプロセッサから順番に実施していき、バックアップされるべきタスクがあるかぎり、次々と負荷の重いプロセッサで実施していく。途中でバックアップされるべきタスクが無くなればそこで処理を終了する。すべてのプロセッサでバックアップを実施してもバックアップされるべきタスクが残っているときはバックアップ依頼手段6120にこの状況を通知して図45のコントローラ間でのバックアップに関する手順を実行させ、本手段は実行を終了する。この図44の手順を内部バックアップ手順と呼び、後にその詳細を具体的な例により説明するが、ここでは本手段の機能を説明するうえでその概略を説明する。

【0112】まず、本手段は手順6510から手順6515にて、ダウンあるいは負荷オーバにより実行できなくなったタスクのバックアップを担当するプロセッサをコントローラ内から決める。この決定において記憶手段6160に格納されている各プロセッサの実行状態に関する情報を用いる。すなわち、多くのタスクをバックアップさせるために負荷の低いプロセッサあるいは余剰性能の大きいプロセッサに決定する。なお、これらの情報

は図48に示した実行分担表と呼ぶ表に格納されている。さらに、ダウンあるいは負荷オーバにより実行できなくなったタスクのうちバックアップされるべきタスクを決定する。この決定では、タスクの属性に関する情報を用い、他のプロセッサでバックアップする必要のないタスクを除外する。このタスクの属性に関する情報は図49に示したタスク負荷表と呼ぶ表に格納されている。

【0113】つづいて本手段は手順6520から手順6540にて、バックアップされるべきタスクにバックアップを担当する部分がバックアップ前に実行していた複数のタスクを追加する。そしてこれらの中から、バックアップを担当するプロセッサがバックアップ後に実行すべきタスクを優先度に応じて選択する。このとき、部分的ダウンあるいは負荷オーバがコントローラ全体のダウンを引き起こさないようにするために、通信手段6110、バックアップ依頼手段6120、スケジューラ6140を提供するタスクなどのコントローラの管理運用に必要な不可欠なタスクを最も優先して選択する。ネットワーク1000の通信量のボトルネックが顕在化して通信不良を引き起こすことがないようにするために、他のコントローラでバックアップした場合に多量の通信を必要とするタスクが次に優先して選択される。その後、その他のタスクを選択する。各優先度においてバックアップを担当するプロセッサの負荷が限界値を越えない範囲で最大負荷となる組合せのタスクを選択する。この負荷の限界値は記憶手段6160に格納されており、限界負荷率と呼ぶ。この選択において記憶手段6160に格納されている各タスクの負荷率とこの選択における優先度とタスクの原籍に関する情報を用いる。タスクの負荷率とは着目したタスクを当該プロセッサで実行した場合の単位時間当りに必要とされる処理時間であり、あらかじめ同等の処理能力を持つプロセッサあるいは当該部分で実行したときに計測した値である。なんらかの事情で計測できないときにはあらかじめ推定し定めた値を用いる。タスクの原籍は各タスクが元々所属するコントローラあるいはその一部分を示す。なお、これらの情報は図49に示したタスク負荷表と呼ぶ表に格納されている。また、この選択において、バックアップを担当するプロセッサがバックアップ前に実行していたタスクであろうとも優先度が低い場合にはバックアップ後に実行させないことがある。このように追い出されたタスクは、バックアップされるべきタスクに追加され、他のプロセッサでバックアップされることを待つ。

【0114】つづいて本手段は手順6550にて、選択したタスクをスケジューラ6140に登録し実行手段6150にて実行を開始させる。バックアップを担当するプロセッサから追い出されたタスクはスケジューラ6140から削除しその実行を停止させる。

【0115】つづいて本手段は手順6560から手順6590にて、バックアップされるべきタスクの状況と、

10

20

30

40

50

コントローラ内でバックアップを担当させていないプロセッサの有無に応じて3通りの処理を実行する。まず、バックアップされるべきタスクに優先度の高いタスクがあり、かつ、コントローラ内の稼働中のプロセッサのすべてにバックアップを担当させる前述の手順を実施していない場合には、次に負荷率が低いプロセッサをバックアップを担当するプロセッサとして選択し、再び手順6520より内部バックアップ手順を繰り返す。多くのタスクあるいは多くの負荷をコントローラ内でバックアップできるように、コントローラ内の各プロセッサに対して順番にバックアップを担当させるというこの繰り返しを設けた。一方、バックアップされるべきタスクが優先度の低いタスクのみの場合と、バックアップされるべきタスクに優先度の高いタスクがあるが、コントローラ内の稼働中のプロセッサのすべてにバックアップを担当させる前述の手順を実施した場合には、コントローラの新しい実行状況をネットワーク1000を通して放送し、バックアップされるべきタスクをコントローラ間でバックアップさせるためにバックアップ依頼手段6120にバックアップされるべきタスクが残っていることを前記接続を介して通知し図45のコントローラ間のバックアップに関する手順を開始させ、本手段の処理を終了する。さらに、一方で、バックアップされるべきタスクがない場合には、コントローラの新しい実行状況をネットワーク1000を通して放送し、本手段の処理を終了する。

【0116】バックアップ依頼手段6120は通信手段6110と記憶手段6160と内部バックアップ手段6170とに接続されている。この手段は、内部バックアップ手段6170からバックアップされるべきタスクが残っていることを通知された場合に、図45の手順に従い、ネットワーク1000に接続された他のコントローラあるいはそのプロセッサからバックアップの依頼先を選定しバックアップを依頼する。すべてのタスクのバックアップが終了するまでこの選定と依頼を繰り返す。この手順をバックアップ依頼手順と呼び、後にその詳細を具体的な例により説明するが、ここでは本手段の機能を説明するうえでその概略を説明する。

【0117】この手段は手順6610にて、記憶手段6160に格納されているすべてのプロセッサの負荷状態に関する情報とを用いて、バックアップを依頼するコントローラあるいはそのプロセッサを選定する。選定基準は他のコントローラのプロセッサのなかで最も負荷の小さいことである。なお、この情報は図49に示した外部受付順位表に格納されている。

【0118】つづいて本手段は手順6620から手順6640にて、さきに選定した依頼先に宛ててバックアップを依頼することを示すメッセージ6600を発信する。このメッセージ6600はバックアップされるべきタスクの情報を含む。メッセージ6600はネットワー

ク1000を介して送られる。そして、依頼先からの応答を待つ。応答は依頼先によってシステム全体に放送されるところの、バックアップの依頼に対する応答であることを示すメッセージ6700である。メッセージ6700はさらに依頼先がバックアップしたタスクとこのバックアップにより依頼先で実行されなくなったタスク

(追い出されたタスク)を示す情報を含む。本手段はこのメッセージ6700を受信するまで待ち状態に入っており、これを受信すると待ち状態が解除されて次の手順6642に進む。

【0119】つづいて本手段は手順6642から6648にて、依頼先がバックアップしたタスクの有無を判定し、依頼先がバックアップしたタスクがある場合にはバックアップの依頼の終了を判定する手順(手順6650から手順6670)に進む。依頼先がバックアップしたタスクがない場合には、さらに手順6644にて他のコントローラのプロセッサのすべてにバックアップを依頼したかを判定する。他のコントローラのプロセッサのすべてにバックアップを依頼していない場合には、手順6646にて他のコントローラのプロセッサの中で今回の依頼先に次いで負荷が小さいものを次のバックアップ依頼先に指定し、手順6620に戻りバックアップの依頼を繰り返す。また、他のコントローラのプロセッサのすべてにバックアップを依頼していた場合には、バックアップが失敗している。そこでこの場合には手順6648に示すように縮退運転手段に縮退運転を開始させ、本手段はバックアップ依頼手順を終了する。

【0120】手順6642の判定にて依頼先がバックアップしたタスクがあった場合には、つづいて、本手段は手順6660から手順6670にて、依頼先からのメッセージ6700に応じて記憶手段6160に格納されている各表を変更する。さらに、依頼先がバックアップしたタスクをバックアップされるべきタスクから削除し、依頼先から追い出されたタスクをバックアップされるべきタスクに追加する。その後、バックアップされるべきタスクの有無を判定し、バックアップされるべきタスクが有るときには手順6610に戻り依頼先の選定とバックアップの依頼を繰り返す。この依頼の繰り返しと依頼先の拒否あるいは受付により、バックアップされるべきタスクを自律的に分散してバックアップすることを可能にした。バックアップ受付手段6130は通信手段6110と記憶手段6160とに接続されている。この手段は、ネットワーク1000を介して他のコントローラから送られてきたバックアップを依頼するメッセージ6600を受け取った場合に、図46の手順に従い、メッセージ6600によってバックアップを依頼されたタスクをバックアップを依頼されたプロセッサの負荷状態に応じてバックアップする。この手順ではバックアップを担当するプロセッサの負荷率が限界負荷率を越えない範囲できる限り大きくなるように、バックアップを依頼され

たタスクとそのプロセッサが実行しているタスクとの入れ替えも実施する。この入れ替えはかなり大きい負荷率のタスクをバックアップするために必要な機能である。大きな負荷率のタスクをバックアップする余裕が無くとも、小さな負荷率のタスクを追い出し余裕を作れば大きな負荷率のタスクをバックアップできることが多い。追い出されたタスクはバックアップされるべきタスクとして、さらに他のプロセッサにバックアップ依頼されそこで実行されることになる。あるいはさらにそこで入れ替えを引き起こすかも知れないが、入れ替えの度により小さな負荷率のタスクが追い出されるためバックアップできる可能性が高まってゆき、最終的にはバックアップできるであろう。この入れ替えにおいて自らのコントローラの管理運用に必要不可欠なタスクあるいは他のコントローラで実行した場合に多量の通信を必要とするタスクを追い出してしまうと、自らのコントローラのダウンやネットワーク 1000 の通信量のボトルネックによる通信の不良を引き起こしたりする。これを防ぐためにバックアップ後に実行するタスクとしてそれらのタスクを優先して選択するこの手順とした。また、本手段はバックアップの結果をシステム全体に知らしめる。この手順をバックアップ受付手順と呼び、後にその詳細を具体的な例により説明するが、ここでは本手段の機能を説明するうえでその概略を説明する。

【0121】この手段は手順 6710 から手順 6730 にて、バックアップされるべきタスクとバックアップを担当するプロセッサで実行されている複数のタスクの中から、バックアップを担当するプロセッサがバックアップ後に実行すべきタスクを優先度に応じて選択する。このとき、バックアップにより自らのコントローラがダウンしないようにするために、通信手段 6110、バックアップ依頼手段 6120、スケジューラ 6140 を提供するタスクなどの自らのコントローラの管理運用に必要不可欠なタスクを最も優先して選択する。ネットワーク 1000 の通信量のボトルネックが顕在化して通信不良を引き起こすことがないようにするために、他のコントローラで実行した場合に多量の通信を必要とするタスクが次に優先して選択される。その後、その他のタスクを選択する。各優先度においてバックアップを担当するプロセッサの負荷が限界値を越えない範囲で最大負荷となる組合せのタスクを選択する。この負荷の限界値は記憶手段 6160 に格納されており、限界負荷率と呼ぶ。この選択は図 44 の内部バックアップ手順の手順 6530 と同一である。また、この選択において、バックアップを担当するプロセッサがバックアップ前に実行していたタスクであろうとも優先度が低い場合にはバックアップ後に実行させないことがある。つづいて、本手段は選択したタスクをスケジューラ 6140 に登録し実行手段 6150 にて実行を開始させる。バックアップを担当するプロセッサから追い出されたタスクはスケジューラ 6140

から削除しその実行を停止させる。

【0122】つづいてこの手段は手順 6740 にて、バックアップするタスクと追い出すタスクを示す情報を含むメッセージ 6700 をネットワーク上に放送（ブロードキャスト）する。このメッセージ 6700 により自らのコントローラの実行状態の変化を他のコントローラのすべてに知らしめる。さらに、本手段はすべてのコントローラは各々の記憶手段 6160、6260、6360、6460 に格納されている実行分担表とタスク負荷表と外部受付順位表をこのメッセージ 6700 によって書き換える。また前述したように、このメッセージ 6700 に基づいて依頼元のバックアップ依頼手段（6220、6320、6420 のいずれか）はバックアップされるべきタスクからバックアップされたタスクを削除し、追い出されたタスクをバックアップされるべきタスクに追加する。

【0123】このように 1 台のダウンあるいは負荷オーバーしたコントローラが実行していた複数のタスクがバックアップを依頼された稼働中のコントローラの負荷に応じて分割されることにより、最終的には複数のコントローラでバックアップされる。つまり、本発明の分散制御システムは 1 台のコントローラのダウンあるいは負荷オーバーを複数台の稼働中のコントローラが自らの負荷状況に応じて分担してバックアップするものである。さらに、このバックアップにおいてネットワークの通信量増加を抑えるために、タスクの通信量に関する属性に応じて分担してバックアップするものである。

【0124】この分散制御システムの物理的な構成を図 47 に示す。コントローラ 1, 2, 3, 4 は同一の構成であり、各々がネットワーク 1000 と接続されたネットワーク接続用コネクタ 41a, 41b, 41c, 41d と、これらのネットワーク接続用コネクタの各々とシリアルデータ用のバス 71a, 71b, 71c, 71d を介して接続された 4 個ずつのプロセッサ（PR1, PR2, PR3, PR4）701~704, 711~714, 721~724, 731~734 と、これらのプロセッサ（PR1, PR2, PR3, PR4）とバス 70a, 70b, 70c, 70d を介して接続されたグローバルメモリ 82a, 82b, 82c, 82d と、これらのプロセッサ（PR1, PR2, PR3, PR4）と接続された入出力処理装置（I/O）83a, 83b, 83c, 83d と、これらのプロセッサに接続され故障の通知を行うバス 70b を含む。各コントローラには DC1, DC2, DC3, DC4 という名前を付けた。各コントローラ内の各プロセッサには PR1, PR2, PR3, PR4 という名前を付けた。ここで、各プロセッサはその中に記したタスクを実行しているものとした。ここでタスク NC1, NC2, NC3, NC4 はネットワーク 1000 を介したコントローラ間のメッセージの送受信を制御するタスクであり、図 43 の通信手段 611

0, 6210, 6310, 6410を提供する。また、タスクCM1, CM2, CM3, CM4は各コントローラ1, 2, 3, 4でのタスクの実行を制御するタスクであり、図43のバックアップ依頼手段6120, 6220, 6320, 6420と、バックアップ受付手段6130, 6230, 6330, 6430を提供する。頭文字がTで始まるタスクは各コントローラが実行する制御演算などであり、図43の制御手段を提供する。OS11などのOSで始まるタスクはバックアップ関係の機能を組み込んだリアルタイムオペレーティングシステムであり、各プロセッサで実行されている。これら各々が図43の内部バックアップ手段6170, 6270, 6370, 6470を提供する。またこれらのリアルタイムオペレーティングシステムの各々が独立に各々のプロセッサに関するスケジューリングを実施している。すなわち、各々が各コントローラのスケジューラ6140, 6240, 6340, 6440を各プロセッサに関する部分に分割して提供している。

【0125】また、各プロセッサ内の故障制御回路6820 (図8, 図10, 図12参照) は他のプロセッサの故障制御回路と協同してバス70bを通信路として用いることにより切り離し手段を提供する。例えば図47のプロセッサ702が故障した場合には、プロセッサ702の故障制御回路が故障を検出し、他のプロセッサ701, 703, 704に対してバス70bを通してプロセッサ702の故障を通知し、同時にプロセッサ702の汎用エンジン(GE)の動作を停止させる。さらに、プロセッサ701, 703, 704の各故障制御回路は各々のプロセッサPRで実行されている内部バックアップ手段を提供するタスクOS11, OS13, OS14に内部バックアップ手順を開始させる。このように図43の切り離し手段6180における機能は物理的には4つのプロセッサに分担されている。

【0126】さらに、図47の各コントローラのなかのすべてのプロセッサがランアダプタ(LA)を持っため、いずれのプロセッサも通信手段6110, 6210, 6310, 6410として動作可能である。従来技術では通信関係の手段が冗長化されておらず信頼性が不充分であったが本実施例の構成により冗長化され信頼性が向上する。各コントローラ内の4つのランアダプタ(LA)のなかで、実際に通信手段6110, 6210, 6310, 6410となるものはタスクNC1, NC2, NC3, NC4が実行されているプロセッサのランアダプタ(LA)である。つまり通信手段として動作中のランアダプタ(LA)が故障して通信が不通になっても、タスクNC1, NC2, NC3, NC4をコントローラ内でバックアップすることで通信を回復することができる。また、従来技術ではプロセッサあるいはプロセッサがローカルメモリを持った場合、そのプロセッサあるいはプロセッサがダウンすると、ローカルメモリに蓄えら

れた制御用の学習結果のデータなどにアクセスできないという問題があった。そこでローカルメモリ(LM)6816をデュアルポートメモリで構成し、バス6819のほかにもバス70aにも接続した。この接続により他のプロセッサからバス71aを介してローカルメモリ(LM)6816へアクセスできるようにした。コントローラ内の4つのプロセッサ(PR1, PR2, PR3, PR4)のローカルメモリ(LM)とグローバルメモリ(GM)によって図34の記憶手段6160, 6260, 6360, 6460を構成している。

【0127】以下では本実施例のバックアップ方法について詳細に例示する。図48は図47の動作状態における実行分担表であり、各記憶手段6160, 6260, 6360, 6460に格納されている。この表の内容はプロセッサを特定可能な名前と、そのプロセッサが実行しているタスクの名称と、それらのタスクによってプロセッサが被る負荷率と、各コントローラ内での各プロセッサに負荷の軽い順に1番から番号を付けた順位(内部負荷順位)である。本例ではコントローラ名とプロセッサ名を連結しプロセッサを特定可能な名称としている。このほかの特定する方式として各々異なる番号を与えるものがある。このプロセッサを特定するために用いる方式は本発明の主旨とは関係なくどのような方式でもよい。本表を用いて内部バックアップ手段6170, 6270, 6370, 6470は、バックアップされるべきタスクをダウンあるいは負荷オーバーしたプロセッサ名から検索し、コントローラ内で負荷の小さいプロセッサから大きいプロセッサへ順番にバックアップを担当させるために内部負荷順位を検索する。また、バックアップ受付手段6130, 6230, 6330, 6440がバックアップを依頼されたプロセッサ名からそのプロセッサが実行しているタスク名を検索するために実行分担表を用いる。なお、内部負荷順位が無くても、内部バックアップ手段はプロセッサ名と負荷率をもとに内部負荷順位と同等の情報を生成し上記の処理を実行可能であるが、バックアップの途中でこの情報を生成するためバックアップを終了するまでの時間が長くなる。バックアップを終了するまでの時間を短縮するために、本実施例では通常の稼働状態の時に内部負荷順位を生成しこの実行分担表に記憶しておく方式にした。

【0128】図49は図47の動作状態におけるタスク負荷表であり、各記憶手段6160, 6260, 6360, 6460に格納されている。この表の内容はタスクの名前と、そのタスクを実行したときの負荷率と、バックアップに関係する属性と、タスクがどのプロセッサのものかを示す原籍である。属性は4つあり、原籍のプロセッサにおいて必要不可欠であるが他のプロセッサでバックアップする必要のない(あるいはしてはならない)タスクであることを示す“fixed”と、コントローラの管理運用に必要なタスクであることを示す“internal

y”と、通信量が多いことを示す“communicative”と、原籍以外のコントローラで実行してもよいタスクであることを示す“somewhere”である。よって、他のプロセッサでバックアップされるべきタスクは属性が“fixed”でないものである。これらのあるプロセッサにおける優先度として扱い優先度が高いものから降順に並べると、最優先のものはこのプロセッサが原籍である“fixed”、次に優先されるものはこのプロセッサが原籍である“internally”、さらに次に優先されるものはこのプロセッサが原籍である“communicative”、優先度の最も低いものはこのプロセッサが原籍である“somewhere”とこのプロセッサが原籍でない“fixed”、“internally”、“communicative”と“somewhere”である。ただし、原籍以外のプロセッサに属性が“fixed”や“internally”のタスクをバックアップさせることはコントローラあるいはシステムの不具合を生じるため、バックアップ手順の中でこれを防いでいる。すなわち、原籍のプロセッサあるいはそれを含むコントローラ（原籍のコントローラと呼ぶ）にて最優先でバックアップされるようにしている。内部バックアップ手段6170、6270、6370、6470と、バックアップ受付手段6130、6230、6330、6440がバックアップをされるべきタスクとバックアップを担当するプロセッサが実行していたタスクとから各々のタスクの優先度と原籍と負荷率を基準としてバックアップ後に実行すべきタスクを選択する時に、タスク名からそれらの内容を知るためにこのタスク負荷表を用いる。

【0129】図50は図47の動作状態における外部受付順位表であり、各記憶手段6160、6260、6360、6460に格納されている。本表はバックアップ依頼手段6120、6220、6320、6420が他のコントローラのプロセッサの中から、最小負荷率のプロセッサの名称を最初に検索し、つづいて必要ならば順番に負荷の大きいものの名称を検索するために用いる。この表はプロセッサ名を負荷率をキーとして昇順に並べたものである。よって、実行分担表に本表の情報も含まれているが、実行分担表で最も負荷率の小さいプロセッサを検索するためには相当な時間がかかる。そこで、検索時間短縮を目的として本表を設けた。本表では先頭のプロセッサ名を読み出すだけですべてのプロセッサの中で最小負荷率のプロセッサ名を知ることができる。それがバックアップを依頼するコントローラに属するときは次の順位を読み出す。この繰り返しで他のコントローラのプロセッサで最も負荷が小さいものを選択できる。本表はダウンあるいは負荷オーバやバックアップの度に書き換えるが、一度並べられているので削除、2分検索、挿入の3ステップでこの表の昇順を維持する。このため、この外部受付順位表を用いた場合にバックアップ時に実行分担表の内容をソーティングを実施する必要がなく処理が高速になる。すなわち、バックアップに要する

時間が短縮できる。

【0130】ここで、実行分担表とタスク負荷表と外部受付順位表の内容の関係を説明する。プロセッサDC1. PR1が実行しているタスクは実行分担表よりタスクOS11, NC1, CM1, T11である。このときの負荷率はタスク負荷表からタスクOS11が5%, NC1が26%, CM1が28%, T11が13%であるからこれらの和である72%となる。実行分担表の例示で省略されたプロセッサの負荷率はこの値より充分大きいとして、プロセッサDC1. PR1の負荷率は昇順で3番目であるから、外部受付順位表では順位が3番目になっている。

【0131】ここで、本実施例のバックアップ手順の理解の助けとして、図51に示すようにコントローラ1のプロセッサ(DC1. PR1)701がダウンした場合において、タスクNC1, CM1, T11が分割してバックアップされる過程とそれにより追い出されたタスクがさらにまた他のプロセッサによってバックアップされる過程を説明する。ここで前述の限界負荷率は90%に定められているものとする。

【0132】(FD0)コントローラ1のプロセッサ701のプロセッサ6811が故障したものとする。この故障により通信手段6110とバックアップ依頼手段6120とバックアップ受付手段6130がダウンし、スケジューラ6140と制御手段6150の各一部分がダウンした。

【0133】(FD1)コントローラ1の切り離し手段6180がプロセッサ701のダウンを検出すると、このプロセッサを停止させる。同時に内部バックアップ手段6170にプロセッサ701のダウンを通知し、内部バックアップ手段6170に図46の内部バックアップ手順を開始させる。

【0134】より詳細には、プロセッサ701の故障制御回路(FD)6820が汎用エンジン(GE)6811の故障を検出すると、バス70bを通して他のプロセッサの故障制御回路(FD)にプロセッサ701の故障を通知する。その後プロセッサ701を停止させる。プロセッサ702, 703, 704の故障制御回路(FD)6820は故障の通知を受けると各々のプロセッサ6811に対してバス70aの割り込み要求信号線を用いて割り込み処理を要求する。この割り込み処理が、内部バックアップ手段6170を提供するタスクOS12, OS13, OS14に内部バックアップ手順を開始させる。なお、停止されたプロセッサのローカルメモリ(LM)6816はバス70aと接続されているため、このプロセッサが停止していても他のプロセッサからアクセス可能である。この接続はタスクをバックアップし再開するときに必要なデータをプロセッサが停止していても取得できるようにしたものである。

【0135】以下、内部バックアップ手段6170が実

行する図46に示した内部バックアップ手順について説明する。

【0136】(OS0)内部バックアップ手段6170が起動される。詳細には稼働中のプロセッサのすべてにおいて内部バックアップを司るタスクが各々独立に図46の内部バックアップ手順を実行する。図51の例示ではプロセッサ(DC1. PR2)702でタスクOS12が内部バックアップ手順を実行する。プロセッサ(DC1. PR3)703でタスクOS13が内部バックアップ手順を実行する。プロセッサ(DC1. PR4)704でタスクOS14が内部バックアップ手順を実行する。バックアップを担当しないタスクが自ら停止し、自動的にバックアップを実施するものが決定する。プロセッサの間にバックアップに関する特定の依存関係がないためコントローラ内を均一のハードウェア及び均一のソフトウェア体形とすることが可能となり、コントローラの構築が容易になる。このように図46の内部バックアップ手順を構築した。次のステップでこの決定方法を説明する。

【0137】(OS1)手順6510:内部バックアップ手段6170は上記の稼働中のプロセッサのなかで負荷が最も小さいプロセッサをバックアップを実施するプロセッサに決定する。

【0138】具体的には前記のタスクの各々が実行分担表からダウンしたプロセッサの内部負荷順位と自らの内部負荷順位を読みだす。ダウンしたプロセッサの負荷が最も小さいのならば内部負荷順位は前述のように1番である。よって、その内部負荷順位が1ならば稼働中のプロセッサの中で負荷が最も小さいものは内部負荷順位が2番目である。この場合には、自らの内部負荷順位が2番であることを判定することにより、自らがバックアップを担当するか否かを定める。内部負荷順位が2番ならばバックアップを担当することになり、手順6520に進む。もし、内部負荷順位が2番でなければ内部バックアップ手順を終了し、バックアップを担当しない。一方、ダウンしたプロセッサの負荷より稼働中のプロセッサの負荷が小さい場合には、稼働中のプロセッサの中で負荷が最も小さいものは内部負荷順位が1番目である。この場合には、自らの内部負荷順位が1番であることを判定することにより、自らがバックアップを担当するか否かを定める。内部負荷順位が1番ならばバックアップを担当することになり、手順6520に進む。もし、内部負荷順位が1番でなければ内部バックアップ手順を終了し、バックアップを担当しない。

【0139】図51の例示ではダウンしたプロセッサ(DC1. PR1)701の内部負荷順位が図48の実行分担表に示すとおり2番であり、プロセッサ(DC1. PR2)702の内部負荷順位が4番であるため、タスクOS12は手順6510で内部バックアップ手順を終了する。また、プロセッサ(DC1. PR3)703の内部負荷順位が3番であるため、タスクOS13は手順

6510で内部バックアップ手順を終了する。一方、プロセッサ(DC1. PR4)704の内部負荷順位は1番であるため、タスクOS14は手順6510から手順6520に進み、内部バックアップ手順を継続する。よって、プロセッサ(DC1. PR4)704がまず初めにバックアップを実施する。

【0140】(OS2)手順6515:内部バックアップ手段6170はダウンにより実行できなくなったタスクのうちバックアップする必要の有るものをタスクキューに登録する。詳細にはまず受付用タスクキューを初期化する。そして、実行分担表からダウンしたプロセッサの名前に対応するタスク名を読みだし、タスク負荷表からそれらのタスクに対する負荷率と属性を読みだす。そして、属性がバックアップする必要がないこと、あるいは、バックアップしてはならないことを示す“fixed”のものを除外して、バックアップされるべきタスクとして受付用タスクキューに登録する。図51の例示ではダウンしたプロセッサ(DC1. PR1)701の名前DC1. PR1に対応するタスク名であるOS11とNC1とCM1とT11が実行分担表から読みだされ、タスク負荷表から各々の負荷率5%, 26%, 28%, 13%と属性“fixed”, “communicative”, “somewhere”が読みだされる。そして、タスクの属性が“fixed”であるタスクOS11に関する情報を除いて、これらの内容が受付用タスクキューに登録される。

【0141】(OS3)手順6520:内部バックアップ手段6170はバックアップを担当するプロセッサが実行しているタスクを受付用タスクキューに追加する。詳細には実行分担表からバックアップを担当するプロセッサの名前に対応するタスク名を読みだし、タスク負荷表からそれらのタスクに対する負荷率と属性を読みだす。そしてそれらのタスクを受付用タスクキューに追加する。同時にそれらの負荷率及び属性も追加する。

【0142】この手順以降はコントローラ内部のすべてのプロセッサがバックアップを担当するまで、あるいは、バックアップされるべきタスクがなくなるまで繰り返し実行される。図51の例示における今回の実行では、バックアップを担当するプロセッサ704の名前DC1. PR4に対応するタスク名であるOS14とT17とT18が実行分担表から読みだされ、タスク負荷表から各々の負荷率5%, 25%, 26%と属性“fixed”, “internally”, “communicative”が読みだされる。これらが受付用タスクキューに追加される。

【0143】(OS4)手順6530:内部バックアップ手段6170はバックアップを担当するプロセッサがバックアップ後に実行すべきタスクを受付用タスクキューから選択する。具体的には、受付用タスクキューをタスクの属性による優先順位と負荷率で並べ変えて、優先順位の高いタスクのなかで限界負荷率に最も近くなる組合せを選択し、さらにすべてを選択しても負荷率に余裕

があるならば次の優先度のタスクのなかでその余裕に最も近くなる組合せを選択する。何れかの優先順位のタスクの中で選択できないものが生じるまで、この組合せと選択を繰り返す。図51の例示では受付用タスクキューが図52に示した内容になり、タスクOS14とNC1とCM1とT17が最終的に選択さる。ここで、バックアップされるタスクはNC1とCM1であり、バックアップされないタスクはT11であり、追い出されるタスクはT18である。

【0144】(OS5) 手順6540: 前手順により選択されたタスクを受付用タスクキューから削除する。具体的には、受付用タスクキューから前手順で選択されたタスクの名称とその負荷率と属性を削除する。図51の例示ではタスクOS14とNC1とCM1とT17が最終的に選択された。当手順では、これらのタスクを受付用タスクキューから削除する。受付用タスクキューに残されたタスクは、バックアップされなかったタスクT11と追い出されたタスクT18である。

【0145】(OS6) 手順6550: 手順6530にて選択されたタスクをスケジューラ6140に登録し実行を開始させる。具体的には、バックアップを担当するプロセッサのスケジューリングを担当するタスクに対してバックアップされるタスクを追加登録し、追い出されるタスクの登録を削除する。図51の例示ではバックアップを担当するプロセッサ704のスケジューリングを担当するタスク、すなわち、リアルタイムオペレーティングシステムであるタスクOS14に、バックアップされるタスクであるタスクNC1とCM1を追加登録し実行を開始させる。また、追い出されるタスクであるタスクT18をスケジューラから削除し実行を停止させる。ここにおいて、プロセッサ701の故障によりダウンしていた通信手段6110とバックアップ依頼手段6120とバックアップ受付手段6130が回復された。以降、コントローラ間のバックアップや制御タスクの実行に必要なネットワーク1000との通信が実行できる。

【0146】(OS7) 手順6560: 当手順はバックアップの状況に応じて内部バックアップ手順を3通りに分岐させる。一つは完全にバックアップが終了したのでバックアップに関する手順を終了させる処理への分岐である。二つは内部でのバックアップを完了しコントローラ間におけるバックアップを開始させる処理への分岐である。三つは内部でのバックアップを繰り返す処理への分岐である。手順6560の中でこれらの分岐を手順6562, 6564, 6566が実施している。手順6562はバックアップすべきタスク、すなわち、受付用タスクキューに残っているタスクの中で優先度の高いタスクが有るか否かを判定し、有るならば処理は手順6564へ進む。無いならば処理は手順6566へ進む。手順6564はコントローラ内のすべてのプロセッサにバックアップを担当させたか否かを判定し、すべてに担当させた

後ならば処理は手順6574以降のコントローラ間におけるバックアップを開始させる処理へ分岐する。すべてに担当させていないならば処理は手順6580以降の内部でのバックアップを繰り返す処理へ分岐する。手順6566はバックアップすべきタスクが残っているか否かを、すなわち、受付用タスクキューにタスクが残っているか否かを判定し、タスクが残っているならば処理は手順6574以降のコントローラ間におけるバックアップを開始させる処理へ分岐する。タスクが残っていないならば処理は手順6572以降のバックアップに関する手順を終了させる処理へ分岐する。

【0147】図51の例示において、今回は、属性が“communicative”であり優先度が高いタスクT11が受付タスクキューに残っており、尚且つ、まだプロセッサ704にバックアップを担当させたのみであるため、手順6580以降の内部でのバックアップを繰り返す処理へ分岐する。

【0148】(OS8) 手順6580: 今回バックアップを担当したプロセッサの次に負荷が小さいものを次のバックアップを担当するプロセッサに選定し、次のバックアップを実施するために手順6520へ戻る。詳細には今回バックアップを担当したプロセッサの内部受付順位とダウンしたプロセッサの内部受付順位を実行分担表から読みだす。前者の値に1を加え次の内部受付順位を求める。この値が後者の受付順位であった場合にはさらに1を加えてダウンしたプロセッサを指定してしまうことを防ぐ。コントローラ内で内部受付順位がこの値であるプロセッサを実行分担表から求める。求めたプロセッサを次回にバックアップを担当するものとして選択する。図51の例示では今回バックアップを担当したプロセッサ704の内部受付順位が1番であり、ダウンしたプロセッサの内部受付順位が2番であるため、次回にバックアップを担当するプロセッサの内部受付順位は3番になる。よって実行分担表においてコントローラ1内で内部受付順位が3番であるプロセッサ703が次のバックアップを担当するものとして選択される。ここから2回目の内部バックアップにはいる。すでに説明した各手順は図51の例示に関する説明のみを記述する。

【0149】(OS9) 手順6520: 図51の例示における今回の実行では、バックアップをプロセッサ703が担当する。そこで、その名前DC1, PR3に対応するタスク名であるOS13とT15とT16が実行分担表から読みだされ、タスク負荷表から各々の負荷率5%, 28%, 41%と属性“fixed”, “somewhere”, “communicative”が読みだされる。これらが受付用タスクキューに追加される。

【0150】(OS10) 手順6530: 図51の例示では受付用タスクキューが図53に示した内容になり、タスクOS13とT15とT16とT11が最終的に選択さる。ここで、バックアップされるタスクはT11で

あり、バックアップされないタスクはT18であり、追
い出されるタスクは無い。

【0151】(OS11) 手順6540: 図51の例示
ではタスクOS13とT15とT16とT11が最終的に
選択された。当手順では、これらのタスクを受付用タ
スクキューから削除する。受付用タスクキューに残された
タスクは、バックアップされなかったタスクT18のみ
である。

【0152】(OS12) 手順6550: 図51の例示
ではバックアップを担当するプロセッサ703のスケジ
ューリングを担当するタスク、すなわち、リアルタイム
オペレーティングシステムであるタスクOS13に、バ
ックアップされるタスクであるタスクT11を追加登録
し実行を開始させる。また、追い出されるタスクが無い
ため、スケジューラから削除され実行を停止させられる
ものは無い。

【0153】(OS13) 手順6560: 図51の例示
において、今回は、属性が“somewhere”であり優先
する必要がないタスクT18のみが受付タスクキューに残
っている。すなわち、この状態は優先度の高いタスクは
残っていないが、バックアップすべきタスクが残ってい
る状態である。よって、当手順は処理を手順6574以
降のコントローラ間におけるバックアップを開始させる
処理へ分岐させる。

【0154】(OS14) 手順6574: 新しい実行状
態を示すメッセージ6700をネットワーク1000に
放送(ブロードキャスト)する。このメッセージにより
他のコントローラはプロセッサのダウンと、そのバック
アップを担当したプロセッサの新しい実行状態を知る
ことができる。このメッセージ6700の内容に合わせて
、各コントローラは各々の記憶手段に格納している実
行分担表とタスク負荷表と外部受付順位表を修正する。
図51の例示において、メッセージ6700はダウン情
報としてプロセッサ701のダウンを示す。さらに、新
しい実行状況としてプロセッサ703が実行しているタ
スクOS13、T15、T16、T11と、プロセッサ
704が実行しているタスクOS14、NC1、CM
1、T17とを示す。修正された実行分担表と外部受付
順位表を図55及び図56に示す。

【0155】(OS15) 手順6590: コントローラ
間でのバックアップを実施するために、バックアップ依
頼手段6120を起動する。詳細には残ったバックアッ
プされるべきタスクを受付用タスクキューからバックア
ップ依頼手段6120が用いる依頼用タスクキューに移
し、受付用タスクキューを空にする。そして、バックア
ップ依頼手段6120にバックアップされるべきタスク
が残っていることを通知する。この通知はリアルタイム
オペレーティングシステムのタスク間通信機能を利用し
て行われてもよい。あるいは、ネットワーク1000を
介した自コントローラから自コントローラ宛のメッセー

ジ通信を利用して行われてもよい。

【0156】(OS16) 以上にて内部バックアップ手
順を終了する。

【0157】続いて、コントローラ間でバックアップ手
順に移行し、タスクT18を他のコントローラでバック
アップする。

【0158】(CM1) コントローラ1のバックアップ
依頼手段6120が起動され、図45のバックアップ依
頼手順の実行を開始する。

【0159】(CM2) 手順6610: バックアップ依
頼手段6120が外部受付順位表を用い、ダウンしたプ
ロセッサを吹く不コントローラとは異なる他のコント
ローラのプロセッサの中で負荷が最も小さいものをバック
アップの依頼先に決定する。詳細には外部受付順位表か
ら順位が1番のプロセッサ名を読みだす。そのプロセッ
サが他のコントローラに所属するならばそこで読みだし
を停止し、そのプロセッサをバックアップの依頼先にす
る。もし、ダウンしたプロセッサと同一のコントローラ
に所属しているならば、再び外部受付順位表から次の順
位のプロセッサ名を読みだす。そして、他のコントロー
ラに所属しているか否かを判定する。他のコントローラ
に所属するプロセッサが読みだされるまで以上を繰り返
す。図51の例示ではダウンしたプロセッサを含むコン
トローラ内でのバックアップを終了しているため、図5
0の外部受付順位表は図56に変更されている。この図
56の外部受付順位表から順位1番のプロセッサ名を読
みだす。この名前はDC2.PR2であり、このプロセッサ
はコントローラDC2に属する。ダウンしたプロセッサ
を含むコントローラはDC1であるため、読みだされた
プロセッサはダウンしたプロセッサを含むコントローラ
とは異なったコントローラに属する。よって次の順位の
プロセッサ名を読みだす処理に戻らずに、このプロセッ
サ(DC2.PR2)712をバックアップを依頼する
プロセッサに決定する。このプロセッサを依頼先と呼
ぶ。

【0160】(CM3) 手順6620: バックアップ依
頼手段6120がバックアップの依頼であることを示す
情報と、依頼用タスクキューに登録されているすべての
タスク名を示す情報とを含むメッセージ6600を作成
し、依頼先に宛てて発信する。このメッセージを依頼メ
ッセージと呼ぶ。図51の例示では依頼メッセージ66
00はタスクT18のバックアップを依頼するものであ
り、依頼先であるプロセッサ712に宛ててネットワー
ク1000を介して送付される。

【0161】(CM4) 手順6630: バックアップ依
頼手段6120はバックアップを依頼したメッセージ6
600に対する回答としてバックアップの依頼先から放
送されるメッセージ6700を待つ。

【0162】以後、メッセージ6700が放送されるま
でバックアップ依頼手段6120はバックアップ依頼手

順を中断して、待ち状態となる。ここからは依頼先のバックアップ受付手段6230の動作について説明する。

【0163】(CM5) 依頼先のプロセッサ712を含むコントローラ2に於て、メッセージ6500を通信手段6210が受け取り、バックアップ受付手段6230を起動する。バックアップ受付手段6230は図46のバックアップ受付手順を開始する。ここで、バックアップ受付手段6230はそれ自身を含むコントローラの記憶手段6260に格納されている受付用タスクキューとタスク負荷表を用いる。

【0164】(CM6) 手順6710: バックアップ受付手段6130がメッセージ6600によって依頼されたタスクとバックアップを依頼されたプロセッサが実行しているタスクを受付用タスクキューに登録する。詳細にはバックアップ受付手段6230はまず記憶手段6260内の受付用タスクキューを初期化する。続いて、メッセージ6600から依頼されたタスクの名称をとりだし、記憶手段6260内のタスク負荷表からそのタスク名に対応する負荷率と属性と原籍を読みだし、先に初期化した受付用タスクキューに登録する。さらに、記憶手段6260内の実行分担表から依頼先のプロセッサの名前に対応するタスク名を読みだし、タスク負荷表からそれらのタスクに対する負荷率と属性を読みだす。そしてそれらのタスクを受付用タスクキューに追加する。同時にそれらの負荷率及び属性も追加する。

【0165】図51の例示では、メッセージ6600によりバックアップを依頼されたタスクの名称T18を受付用タスクキューに登録する。同時に、タスク負荷表からその名称に対応した負荷率26%と属性“somewhere”と原籍DC1. PR4を読みだしこれらに登録する。さらに、バックアップを依頼されたプロセッサ712が実行しているタスクの名称として、そのプロセッサ名DC2. PR2に対応するタスク名OS22とT21とT22を実行分担表から読みだす。さらに、タスク負荷表から各々のタスクに対応した負荷率5%, 18%, 35%と属性“fixed”, “communicative”, “somewhere”と原籍DC2. PR2, DC2. PR2, DC2. PR2とを読みだす。そして、これらのタスク名と負荷率と属性と原籍を受付用タスクキューに追加する。

【0166】(CM7) 手順6720: バックアップ受付手段6130はバックアップを依頼されたプロセッサがバックアップ後に実行すべきタスクを受付用タスクキューから選択する。具体的には、受付用タスクキューをタスクの属性による優先順位と負荷率で並べ変えて、優先順位の高いタスクのなかで限界負荷率に最も近くなる組合せを選択し、さらにすべてを選択しても負荷率に余裕があるならば次の優先度のタスクのなかでその余裕に最も近くなる組合せを選択する。何れかの優先順位のタスクの中で選択できないものが生じるまで、あるいは、全てのタスクが選択されるまでこの組合せと選択を繰り返す。

返す。図51の例示では受付用タスクキューが図54に示した内容になり、タスクOS22とT21とT23とT18が最終的に選択さる。ここで、バックアップされるタスクはT18であり、バックアップされないタスクは無い、追い出されるタスクも無い。すなわち、受付用タスクキュー内の全てのタスクが選択された。

【0167】(CM8) 手順6730: バックアップ受付手段6130は前手順により選択されたタスクをスケジューラ6240に登録し、それらの実行を開始する。

10 具体的には、バックアップされるタスクをスケジューラ6240に追加して登録し、その実行を開始させる。また追い出されるタスクの登録をスケジューラ6240から削除し、その実行を停止させる。図51の例示ではタスクT18を登録し、実行を開始させる。登録を削除するものはない。

【0168】(CM9) 手順6740: バックアップ受付手段6130はバックアップを依頼されたプロセッサの新しい実行状態を示すメッセージ6700をネットワーク1000に放送(ブロードキャスト)する。このメッセージ6700はバックアップの依頼に対する回答であることを示す情報と、バックアップ受付手順が終了したときにバックアップを依頼されたプロセッサが実行しているタスクの情報と、バックアップを依頼されたタスクの中でバックアップしたタスクを示す情報と、バックアップを依頼されたプロセッサから追い出されてその実行を停止されたタスクを示す情報を含む。このメッセージにより他のコントローラはバックアップを依頼されたプロセッサの新しい実行状態を知り、各々の記憶手段に格納している実行分担表とタスク負荷表と外部受付順位表を修正する。また、このメッセージ6700によりバックアップの依頼元であるバックアップ依頼手段6120は待ち状態から抜け出し、バックアップ依頼手順を再開する。図51の例示において、メッセージ6700は新しい実行状況としてプロセッサ712が実行しているタスクOS22, T21, T22, T18を示す。バックアップされたタスクとしてタスクT18を示す。追い出されたタスクとしては、該当するものがないため、追い出されたものがないことを示す。

【0169】(CM10) バックアップ受付手段6230がバックアップ受付手順を終了する。

【0170】以降、メッセージ6700により処理を再開したバックアップ依頼手順6120の処理について説明する。

【0171】(CM11) 手順6642: バックアップ依頼手段6120がメッセージ6700からバックアップされたタスクを示す情報をとりだす。そして、バックアップされたタスクがあるか否かを判定する。もし、バックアップされたタスクがないならば手順6644にて全てのプロセッサにバックアップを依頼したか否かを判定する。さらにもし、すべてのプロセッサに依頼していない

ならば手順6646にて外部負荷順位表を参照して、今回の依頼先に次いで負荷が小さいプロセッサを依頼先に選定する。そして、再び手順6620以降によりバックアップの依頼を繰り返す。手順6644にて、もし全てのプロセッサに依頼した後であったならば、バックアップが失敗したため、縮退運転手段を起動し縮退運転を行う。さて、手順6642に戻る。ここで、バックアップされたタスクが存在するならば手順6650へ進む。

【0172】図51の例示ではタスクT18がバックアップされたため、手順6650へ進む。

【0173】(CM12) 手順6650: バックアップ依頼手段6120はメッセージ6700の内容に応じて記憶手段6160内の実行分担表とタスク負荷表と外部受付順位表を変更する。図51の例示では図55の実行分担表のマイクとコンピュータDC2. PR2の実行しているタスクの欄にタスクT18を追加する。同じく負荷率の欄を58%の負荷率から84%に変更する。コントローラ2 (DC2) 関係の内部負荷順位を変更し、プロセッサDC2. PR1を2番に、プロセッサDC2. PR2を4番に、プロセッサDC2. PR3を1番に、プロセッサDC2. PR4を3番にする。外部受付順位表では受付順位1番のプロセッサDC2. PR2が負荷率84%となったためそれを後ろに回し、負荷率84%以下のプロセッサの順位を1番ずつ繰り上げる。したがって、外部受付順位が1番のプロセッサはDC3. PR1になる。

【0174】(CM13) 手順6660: バックアップ依頼手段6120は依頼用タスクキューからメッセージ6700に示されたところのバックアップされたタスクを削除する。また、依頼用タスクキューにメッセージ6700に示されたところの追い出されたタスクを追加する。この操作は1回のバックアップ依頼で全てのバックアップが終了しなくてもよいように設けた。すなわち、1回のバックアップでバックアップできなかったタスク、あるいは、負荷率を上げるために追い出されたタスクがある場合には、依頼用タスクキューにタスクが残り、これを今回の依頼先と異なった依頼先に次回に依頼できるようにしたものである。図51の例示では依頼用タスクキューからタスクT18が削除され、依頼用タスクキューが空になる。

【0175】(CM14) 手順6670: バックアップ依頼手段6120は依頼用タスクキューにタスクが残っているか否かを判定する。もし残っているならば再び手順6610に戻り新規に依頼先を決定し、バックアップを依頼する。また、もし、依頼用タスクキューにタスクが残っていないならば、全てのタスクのバックアップを終了したことになる。このときバックアップ依頼手段6120はバックアップ依頼手順の終了に進む。

【0176】(CM15) バックアップ依頼手段6120はバックアップ依頼手順を終了する。

【0177】以上のステップにより本実施例の分散制御システムは1台のプロセッサ(DC1. PR1) 701のダウンに対して、そこで実施されていたタスクNC1, CM1, T11を自律的に分割してそれぞれをコントローラ1のプロセッサ(DC1. PR4) 704, コントローラ1のプロセッサ(DC1. PR3) 703, コントローラ2のプロセッサ(DC2. PR2) 712においてバックアップする。図51の例示ではダウンしたプロセッサを持つコントローラと、バックアップを依頼するコントローラが同じ場合を示した。しかし、本発明の主旨によればコントローラ1、2、3、4の各々がすべてのコントローラの実行状態を記述している実行分担表を持っているために、あるコントローラのダウンにおいて他の一つのコントローラがさらに他のもう一つのコントローラに対してバックアップを依頼することが可能である。具体的には図51の例示において、コントローラ1のプロセッサ704でバックアップされたタスクCM1によって提供されるバックアップ依頼手段6120の代わりに、コントローラ2のプロセッサ711で実行されているタスクCM2によって提供されるバックアップ依頼手段6220が記憶手段6260に格納された実行分担表と外部受付順位表を用いてバックアップを依頼することもできる。このために、実行分担表とタスク負荷表と外部受付順位表は全てのコントローラ、あるいは、全てのプロセッサの状態を記憶している。図51の例示ではダウンを検出したコントローラのバックアップ依頼手段が作動するものとした。

【0178】本実施例はコントローラが4台のシステムであるが、本発明の主旨を逸脱しない範囲で2台以上のコントローラを持つシステムに拡張できる。さらに、本実施例では1台のコントローラが4個のプロセッサを含むものとしたが本発明の主旨を逸脱しない範囲で1個以上のプロセッサを持つシステムに拡張できる。

【0179】

【発明の効果】一つのコントローラを数値演算機能はもとよりニューロ、ファジィ、行列等を高速に演算するための並列演算機能を実行するプロセッサ、シーケンス制御演算を高速に実行するプロセッサ、さらには、ネットワークとの通信機能を実行するプロセッサ等の複数のプロセッサから構成し、各機能を並列に実行可能とすることにより、幅広い分野に適用可能なコントローラが実現できる。このコントローラをネットワークワーク接続することにより、幅広い分野に適用可能分散制御システムが実現できる。

【0180】また、入出力及びプロセッサを機能プログラムできるチップを実現する方法も提供した。このことにより、多種多様な制御対象に対しても少ない種類のコントローラで対応でき、保守性の点で優れたシステムを構成できる。

【0181】各プロセッサをより小さい単位のプロセッ

サエレメントで構成することにより、フレキシビリティが向上しLSI化も容易になり、コントローラの小型化を図れる。従って、コントローラ本体の小型化が図れ、分散設置可能なコントローラが実現できる。また予備プロセッサ及び不揮発性メモリの予備領域を設けたことにより、部分故障に対してコントローラ自身で自己修復が可能となる。

【0182】また、バスの拡張が容易なバスコネクタつきの外観構造を採ることで小規模から大規模まで広い範囲の制御対象への対応が可能となる。

【0183】また、ユーザの書いたプログラムとニューラルネットワーク等の複数の出力生成手段によって得られる複数の出力を、学習状況や経時変化等に対し適応的に複合させてより良い制御を可能にすると共に、簡単な操作でユーザの好みの制御を行わせることができる。

【0184】さらに、本発明により分散制御システムにおけるコモンモード故障時に異常情報が多数ネットワークに発生し通信ネックが発生することを防止することができる。また、分散制御システムにおける故障箇所の設定が可能である。

【0185】さらにまた、本発明により分散制御システムのコントローラの多重ダウンに対しても負荷を自律的に稼働中のコントローラに分散しバックアップすることが可能である。さらにまた、本発明により分散制御システムのコントローラを複数のプロセッサで構成したため、同一コントローラ内のプロセッサの多重ダウンに対しても負荷を自律的に稼働中のプロセッサに分散しバックアップすることが可能である。このため、信頼性及び耐故障性を高めることが可能である。

【図面の簡単な説明】

【図1】本発明の一実施例であるところのプラント制御システム。

【図2】従来の鉄鋼圧延制御システム。

【図3】具体的な本発明の応用例であるところの鉄鋼圧延制御システム。

【図4】別の具体的な本発明の応用例であるところの自動車の制御システム。

【図5】別の具体的な本発明の応用例であるところの自動車の制御システム。

【図6】本発明を階層システムに適用した場合のシステム構成例。

【図7】本発明の一実施例であるところのコントローラの内部構成図。

【図8】本発明の一実施例であるところのコントローラの内部構成図。

【図9】本発明の他の一実施例であるところのコントローラの内部構成図。

【図10】本発明の他の一実施例であるところのコントローラの内部構成図。

【図11】本発明のさらに他の一実施例であるところの

コントローラの内部構成図。

【図12】本発明のさらに他の一実施例であるところのコントローラの内部構成図。

【図13】機能プログラム可能なコントローラの全体ブロック図。

【図14】入出力間の接続を不揮発性メモリを用いてプログラマブルとしたスイッチアレイの構成例。

【図15】図10に示したスイッチアレイの中のメモリセルの構成例。

10 【図16】スイッチアレイに予備領域を設けた実施例。

【図17】マイクロプログラムメモリに不揮発性メモリを用いたプロセッサの内部構成例。

【図18】各プロセッサをプロセッサエレメントの組合せで構成した実施例。

【図19】I/O83の内部構成図。

【図20】I/O83を1チップの半導体で実現した構成図。

【図21】I/O83を1チップの半導体で実現した構成図。

20 【図22】本発明のコントローラ1'を1チップの半導体で実現した構成図。

【図23】本発明のコントローラ1'を1チップの半導体で実現した構成図。

【図24】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図25】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図26】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

30 【図27】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図28】本発明における制御演算部の基本構成図。

【図29】本発明における制御演算部の全体概要図。

【図30】出力計算部の構成。

【図31】評価部の構成。

【図32】複数の評価基準を有する場合の全体概要図。

【図33】ニューラルネットによって重み値を生成する場合の基本構成図。

【図34】診断機能付きの分散型コントローラ構成図。

40 【図35】分散システムの各コントローラの制約条件の例図。

【図36】各コントローラの通信インタフェースの構成図。

【図37】異常情報フォーマット図。

【図38】制約未充足時の異常信号送信処理の処理手順図。

【図39】不要情報判定アルゴリズムの処理手順図。

【図40】診断情報取込み判断処理手順図。

【図41】故障情報の加工処理手順図。

50 【図42】異常情報の包含関係を示す例図。

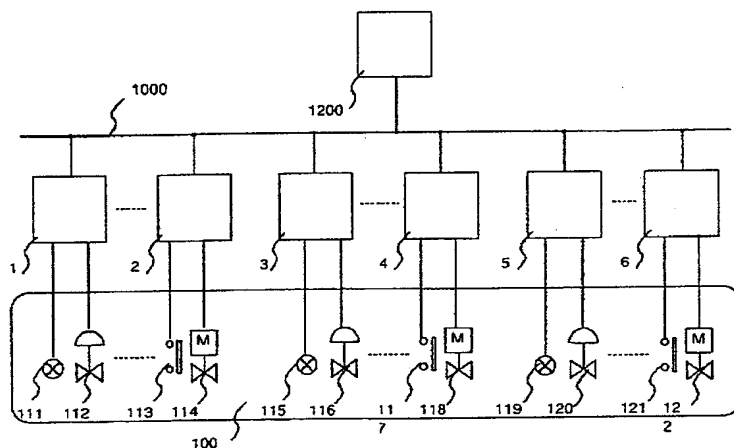
- 【図 43】 自律分散制御システムの論理的構成図。
 【図 44】 自律的なバックアップ依頼手順図。
 【図 45】 自律的なバックアップ受付手順図。
 【図 46】 内部バックアップ手順図。
 【図 47】 自律分散制御システムの物理的構成図。
 【図 48】 実行分担表を示す例図。
 【図 49】 タスク負荷表を示す例図。
 【図 50】 外部受付順位表を示す例図。
 【図 51】 自律分散制御システムのバックアップ方法を
 示す例図。
 【図 52】 1 回目の内部バックアップにおける受付用タ
 スクキューの例図。
 【図 53】 2 回目の内部バックアップにおける受付用タ
 スクキューの例図。
 【図 54】 1 回目の外部バックアップにおける受付用タ
 スクキューの例図。
 【図 55】 内部バックアップ後の実行分担表を示す例
 図。
 【図 56】 内部バックアップ後の受付順位表を示す例
 図。

【符号の説明】

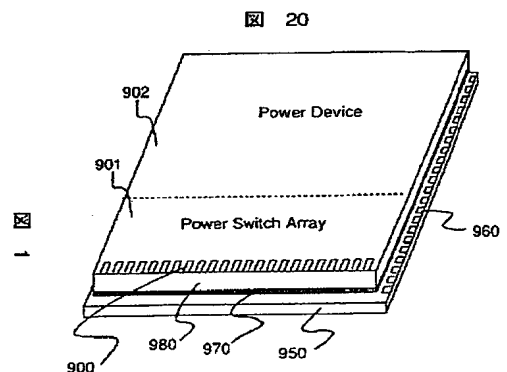
1, 2, 3, 4, 5, 6, 6000…コントローラ、4
 0…NCP、41a, 41b, 41c, 41d…ネット
 ワーク接続用コネクタ、52, 82, 82a, 82b,
 82c, 82d…グローバルメモリ、53…I/O、5
 4, 56, 58, 84, 87, 90, 93…プロセッ
 サ、55, 57, 59, 85, 88, 91, 94…ロー
 カルメモリ、60, 61, 62, 63…バス拡張用コネ
 クタ、64…I/Oコネクタ、70a, 70b, 70
 c, 70d…バス、71a, 71b, 71c, 71d… 30
 シリアルデータ通信用のバス、72a, 72b, 72
 c, 72d…故障通知用のバス、83a, 83b, 83
 c, 83d…入出力処理装置 (I/O)、86, 89, *

* 92, 95…ランアダプタ、100…プラント、70
 1, 702, 703, 704…コントローラ 1 のプロセ
 ッサ、711, 712, 713, 714…コントローラ 2
 のプロセッサ、721, 722, 723, 724…コン
 トローラ 3 のプロセッサ、731, 732, 733, 7
 34…コントローラ 4 のプロセッサ、1000…ネット
 ワーク、2001…センサ等からの入力データ、
 2002…入力データ、2003…出力データ、200
 4…アクチュエータ等への出力データ、2010…セン
 サ等、2011…入力処理部、2012…出力計算部、
 2013…出力処理部、2014…アクチュエータ等、
 2015…評価部、1200…マンマシン装置、201
 7…タッチパネル、マウス等のポインティングデバイ
 ス、2021…重み付け部、2022…出力データ合成
 部、2101…ニューラルネットワーク、2102…出
 力生成手段 2、2103…出力生成手段 3、2122…
 重み値更新信号、6110, 6210, 6310, 64
 10…通信手段、6120, 6220, 6320, 64
 20…バックアップ依頼手段、6130, 6230, 6
 330, 6430…バックアップ受付手段、6140,
 6240, 6340, 6440…スケジューラ、615
 0, 6250, 6350, 6450…制御手段、616
 0, 6260, 6360, 6460…記憶手段、617
 0, 6270, 6370, 6470…内部バックアップ
 手段、6180, 6280, 6380, 6480…切り
 離し手段、6811…汎用エンジン (GE)、6812
 …ニューラルエンジン (NE)、6813…シーケンス
 エンジン (SE)、6814…ランアダプタ (LA)、
 6815…フリーランタイマ (FRT)、6816…ロー
 カルメモリ (LM)、6817…バスインタフェース
 (BIF)、6818…ダイレクトメモリアクセス制御
 (DMAC)、6819…プロセッサ内のバス、682
 0…故障制御回路。

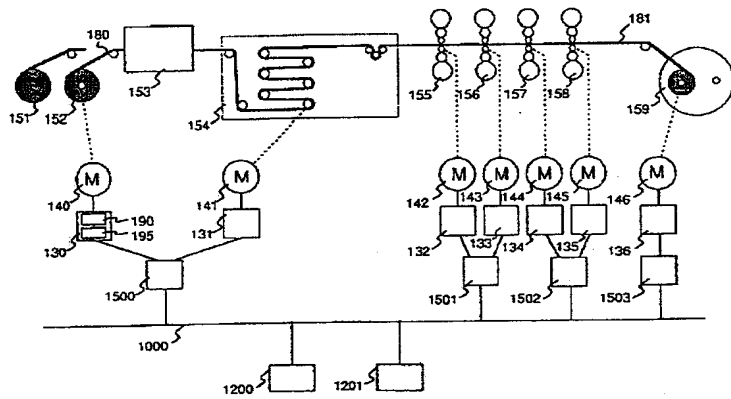
【図 1】



【図 20】

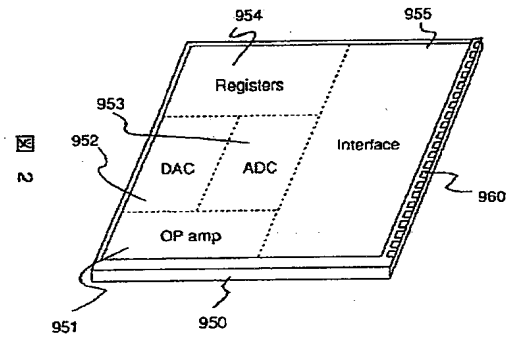


【図2】

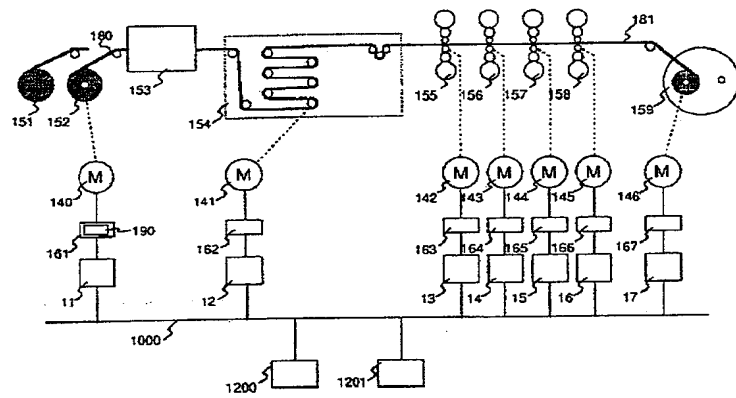


【図21】

図 21

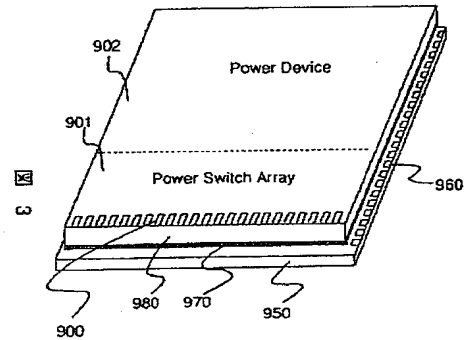


【図3】

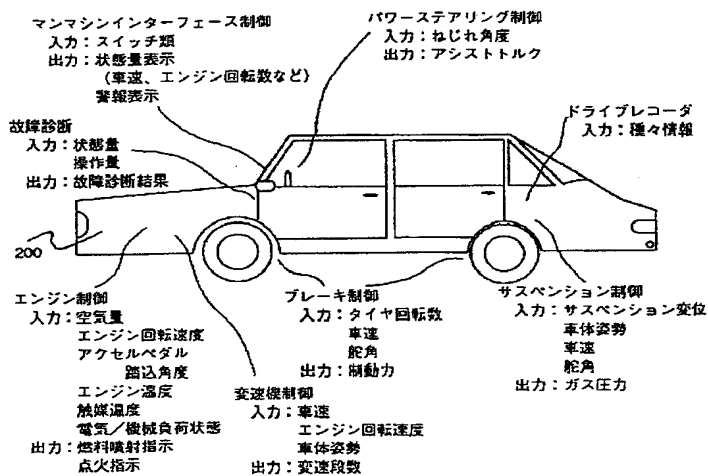


【図22】

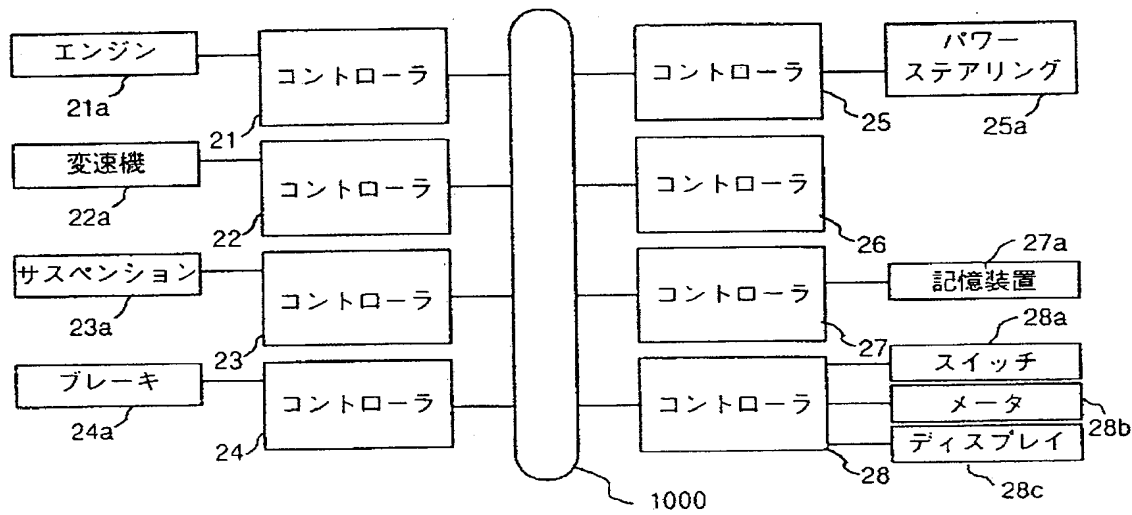
図 22



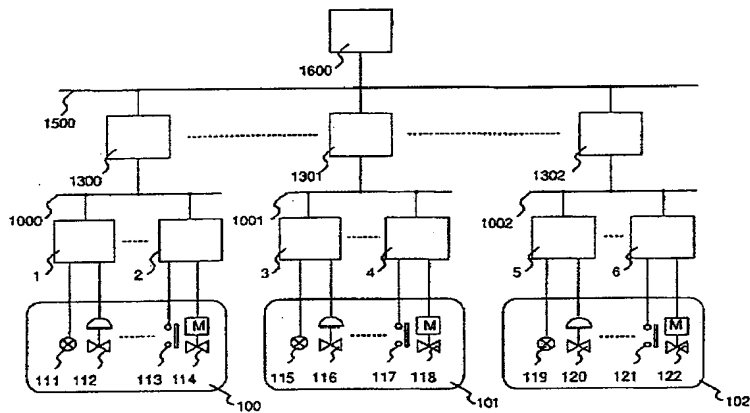
【図4】



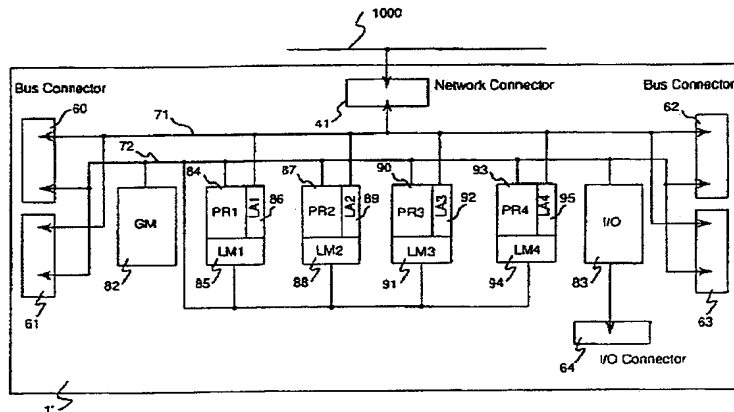
【図 5】



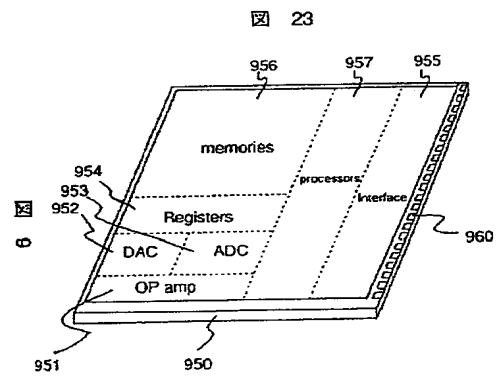
【図 6】



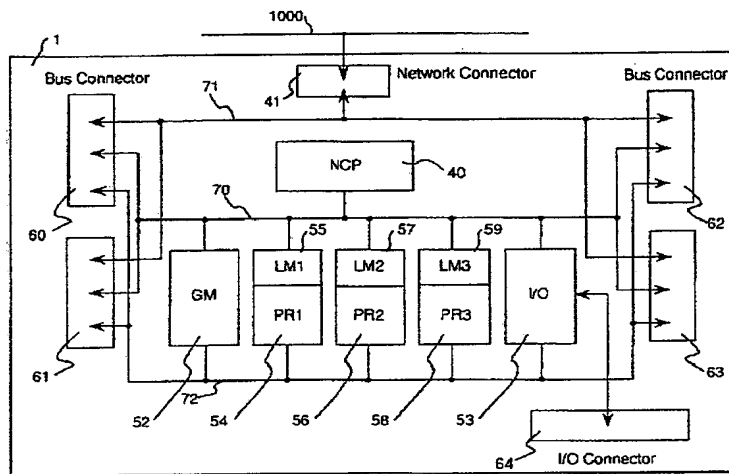
【図 9】



【図 23】



【図7】

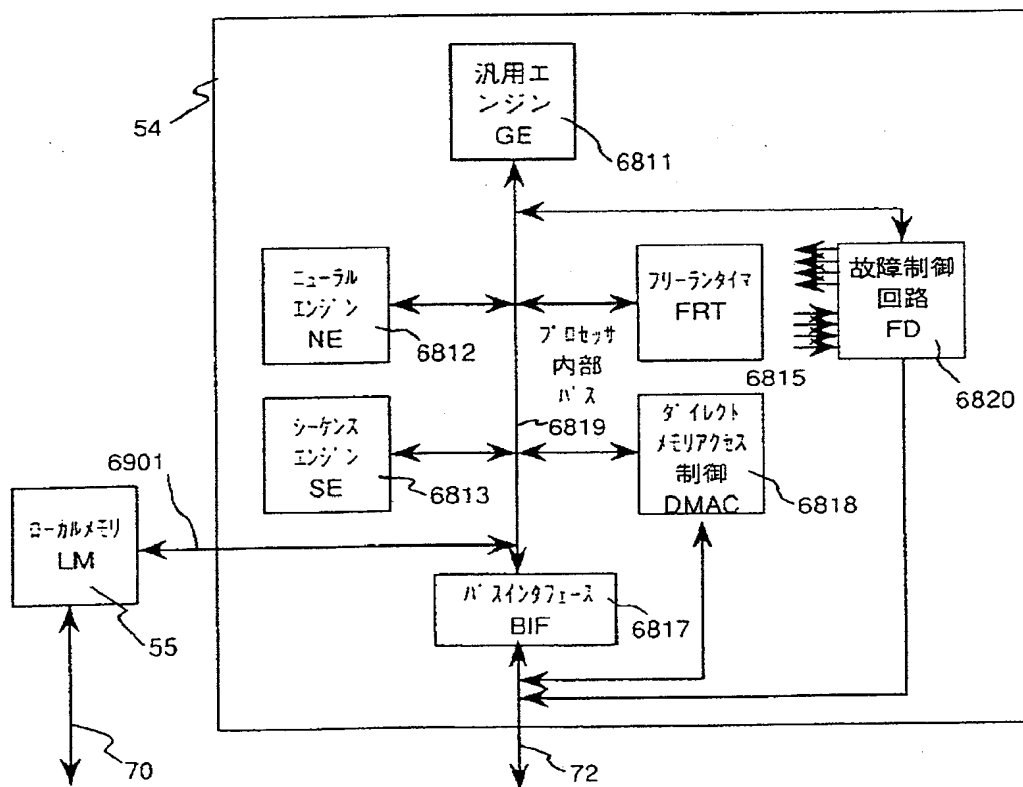


【図50】

図 50

外部受付順位表	
順位	マイコンユニット
1	DC1.PR4
2	DC2.PR2
3	DC3.PR1
4	DC2.PR3
5	DC3.PR3
6	DC2.PR1
以下省略	
.	.
.	.

【図8】



【図10】

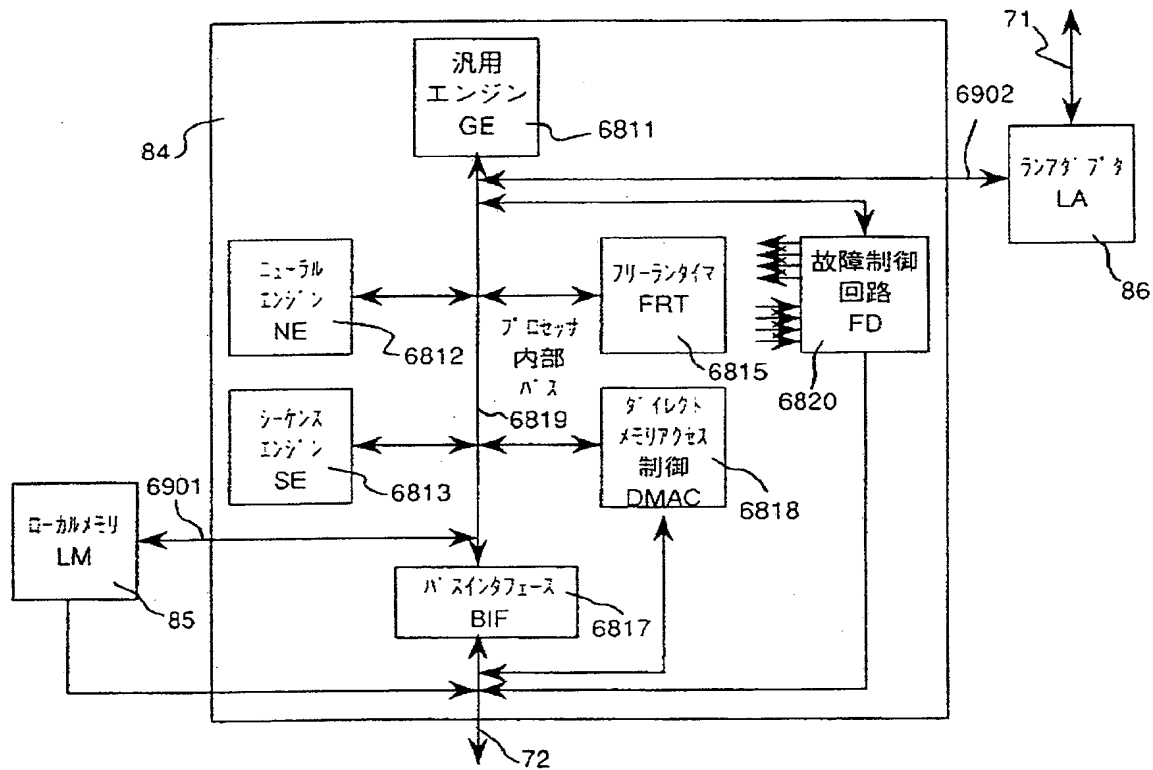
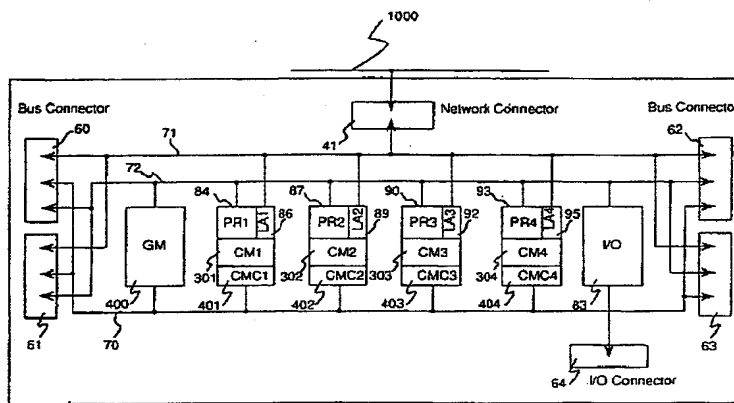


図 10

【図11】



【図54】

図 54

受付用タスクキュー				
タスク	負荷率 (%)	属性	原籍	選択結果
OS22	5	fixed	DC2.PR2	選択
T21	18	communicative	DC2	選択
T18	26	somewhere	—	選択
T22	35	somewhere	—	選択

【図25】

図 25

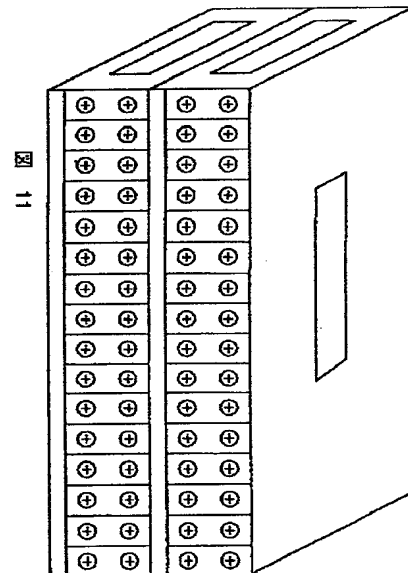
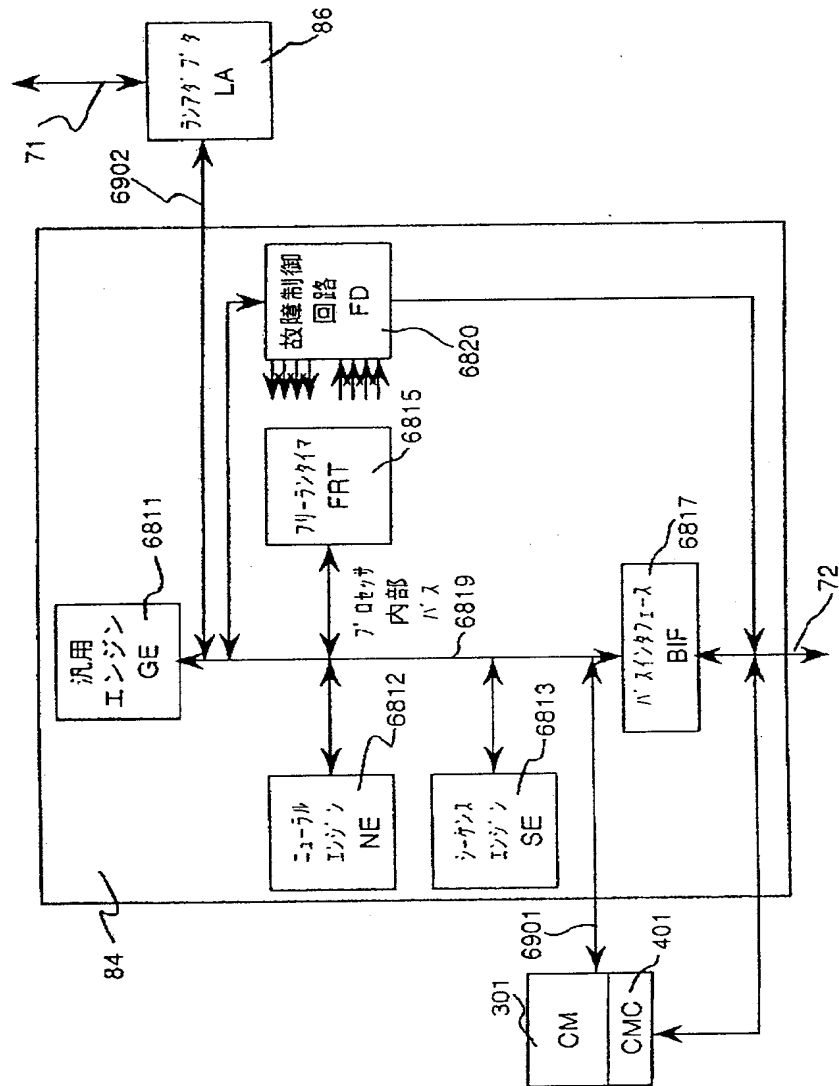


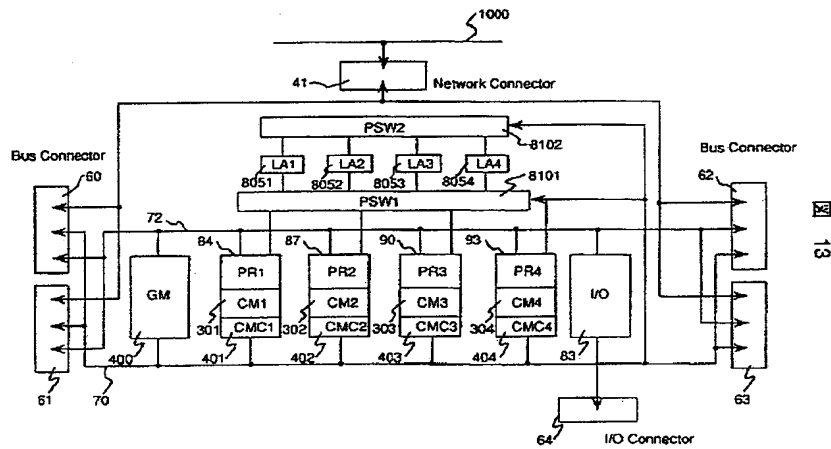
図 11

【図12】

図 12

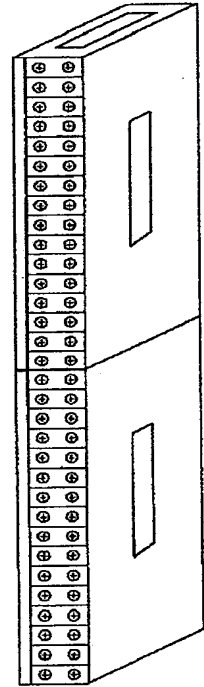


【図13】



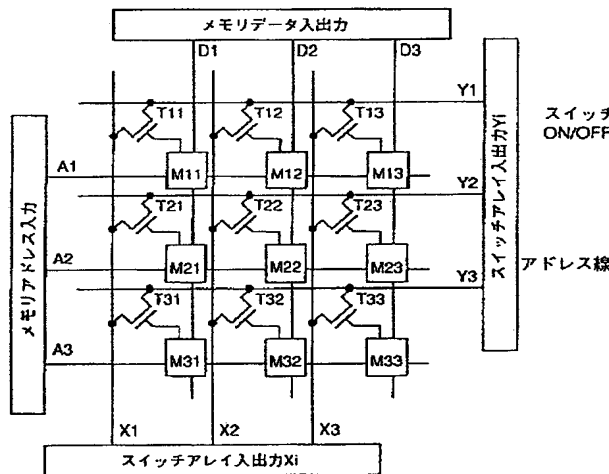
【図26】

図 26



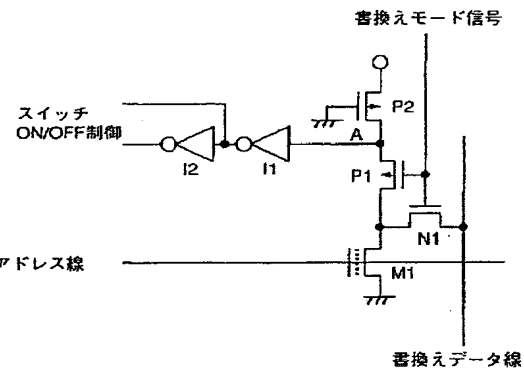
【図14】

図 14



【図15】

図 15

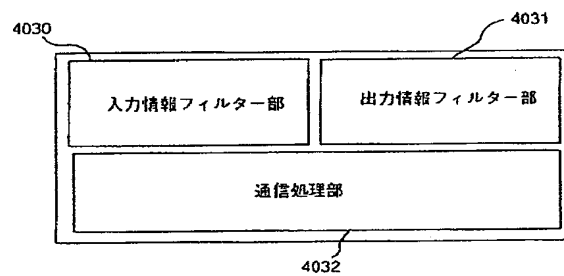
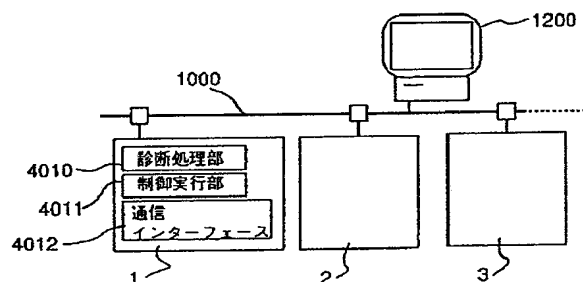


【図36】

図 36

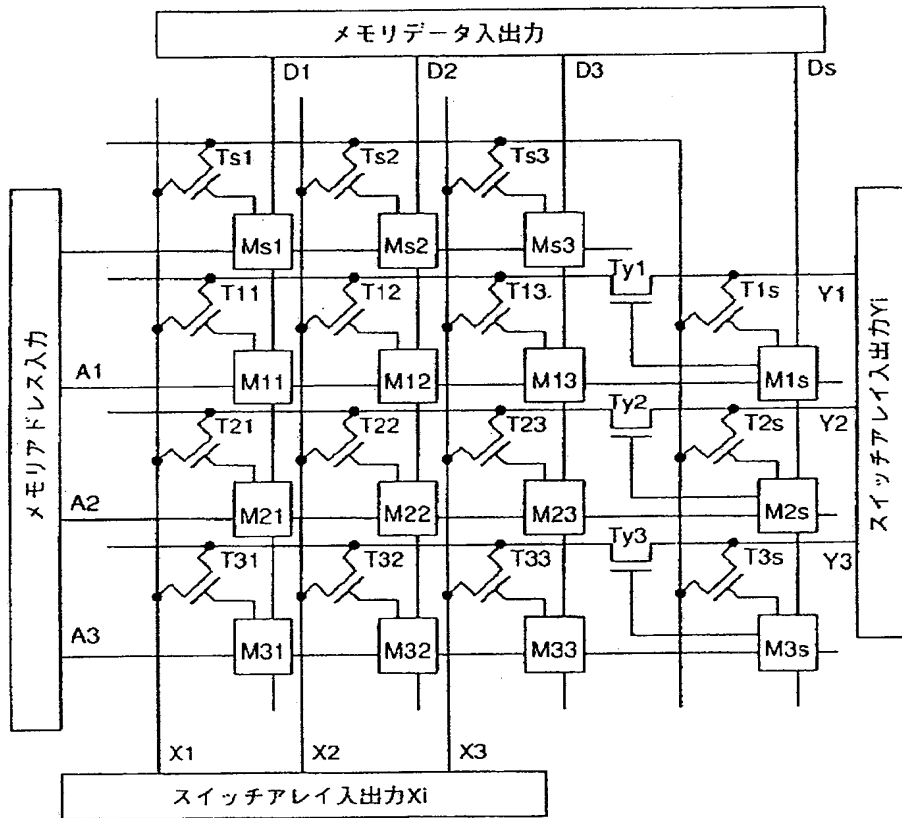
【図34】

図 34



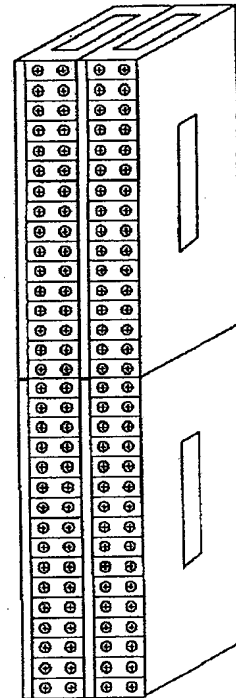
【図16】

図 16

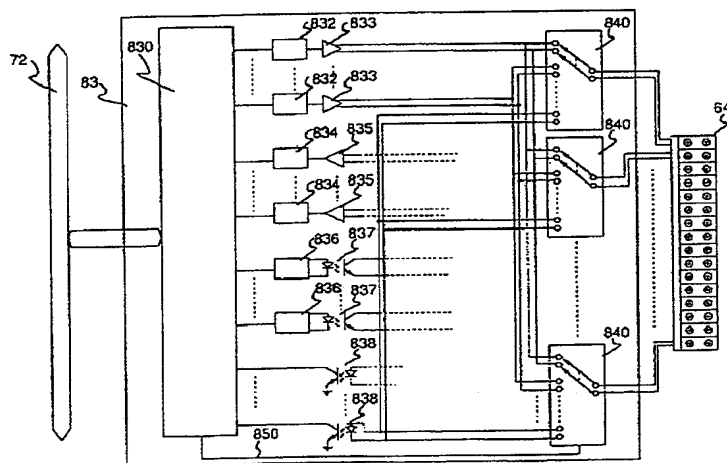


【図27】

図 27



【図19】



【図56】

図 56

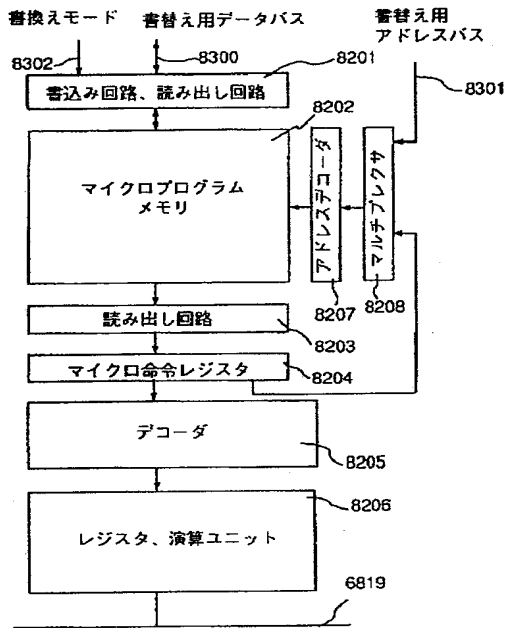
順位	177012 1-2
1	DC2.PR2
2	DC3.PR1
3	DC2.PR3
4	DC3.PR3
5	DC2.PR1
6	DC3.PR2
-	以下省略
-	-

図 19

左記の様に網掛けした部分が更新されている

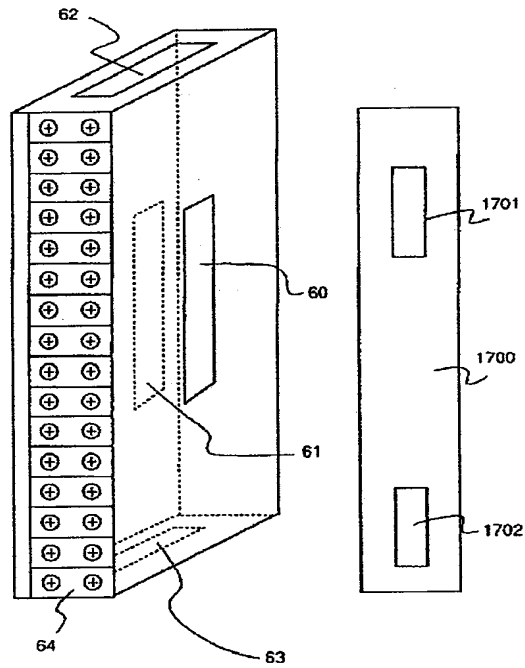
【図17】

図 17



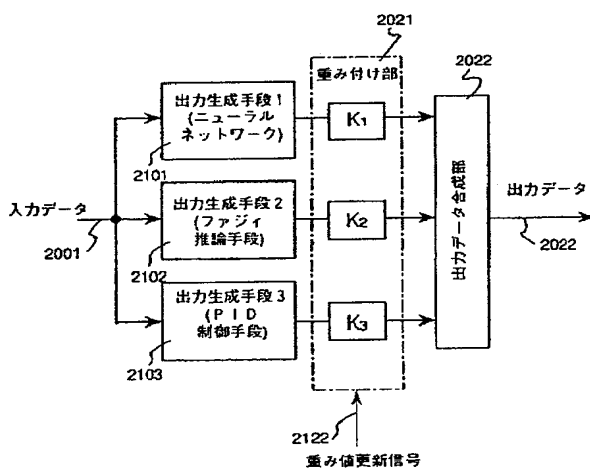
【図24】

図 24



【図28】

図 28



【図29】

図 29

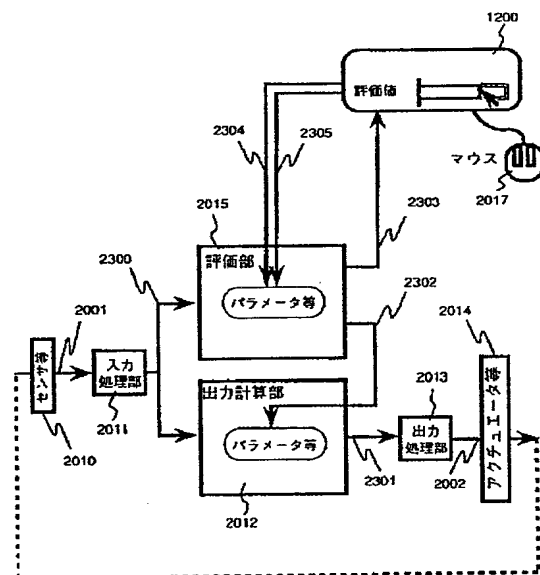
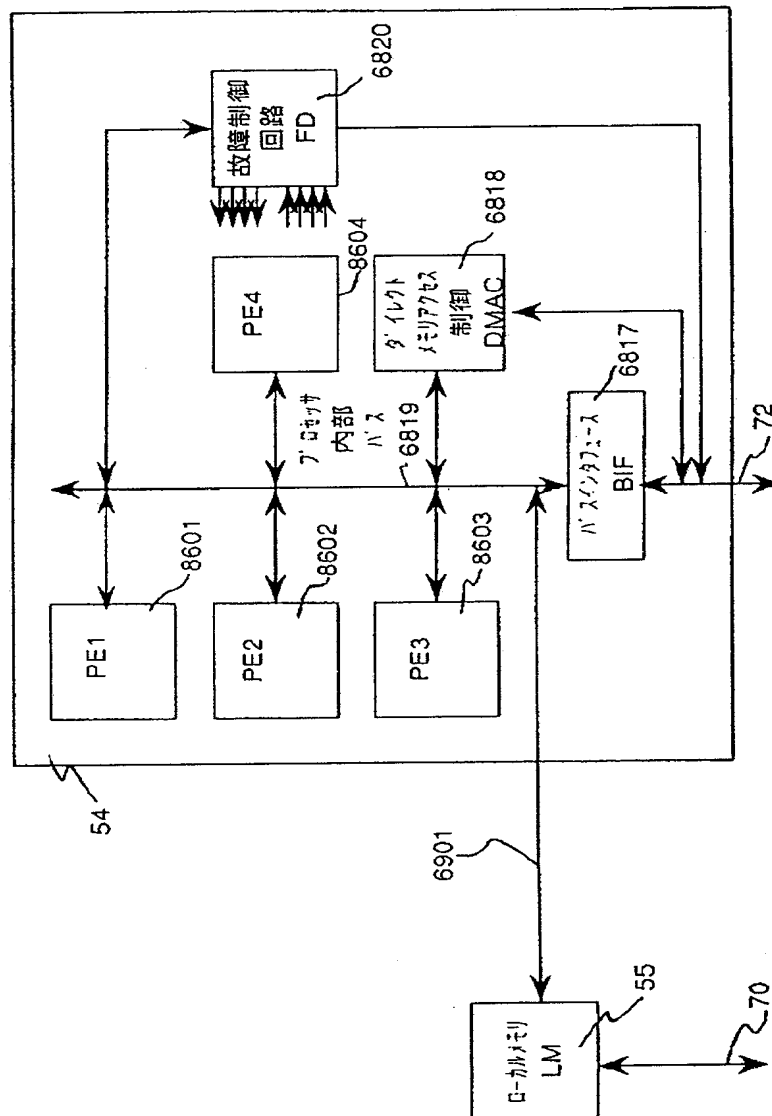


图 18



30 ㊦

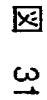
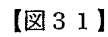
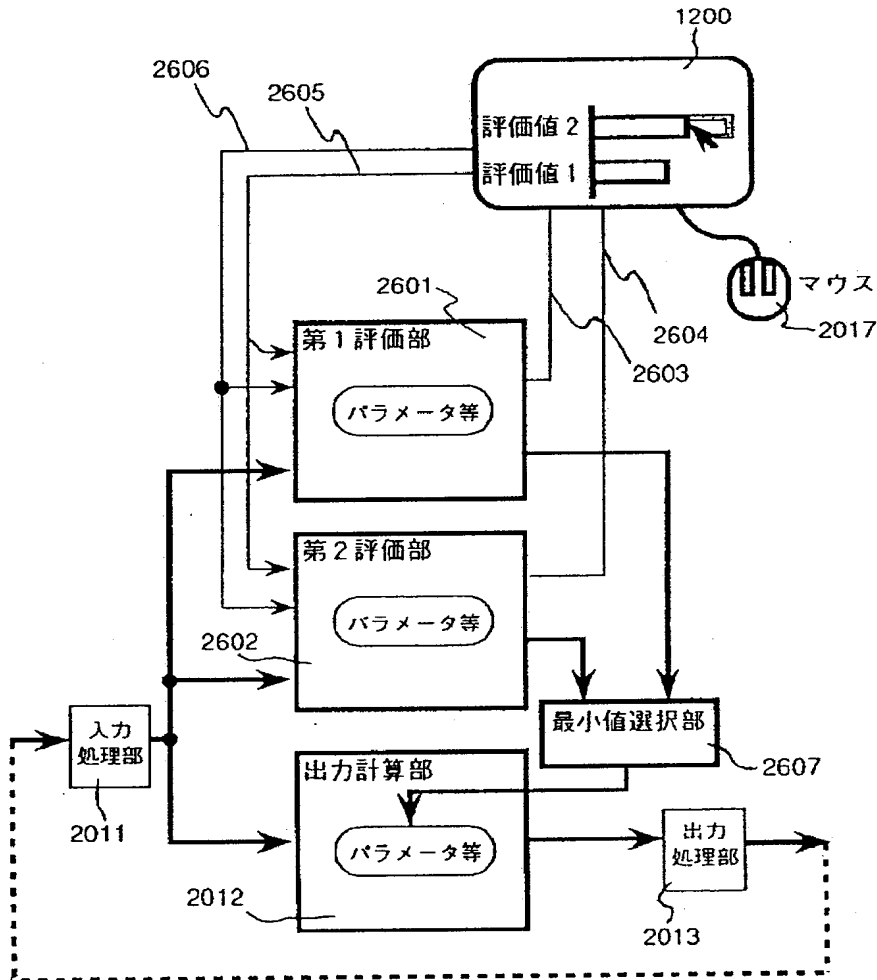


图 52

受付用タスクキュー				
タスク	負荷率 (%)	属性	原籍	選択結果
OS14	5	fixed	DC1.PR4	選択
NC1	26	internally	DC1	選択
CM1	28	internally	DC1	選択
T11	13	communicative	DC1	非選択
T17	25	communicative	DC1	選択
T18	26	somewhere	—	追いつ出し

【図32】

図 32



【図37】

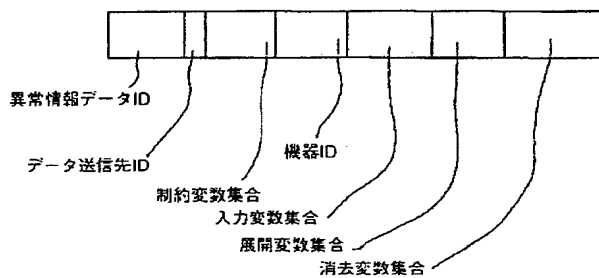


図 37

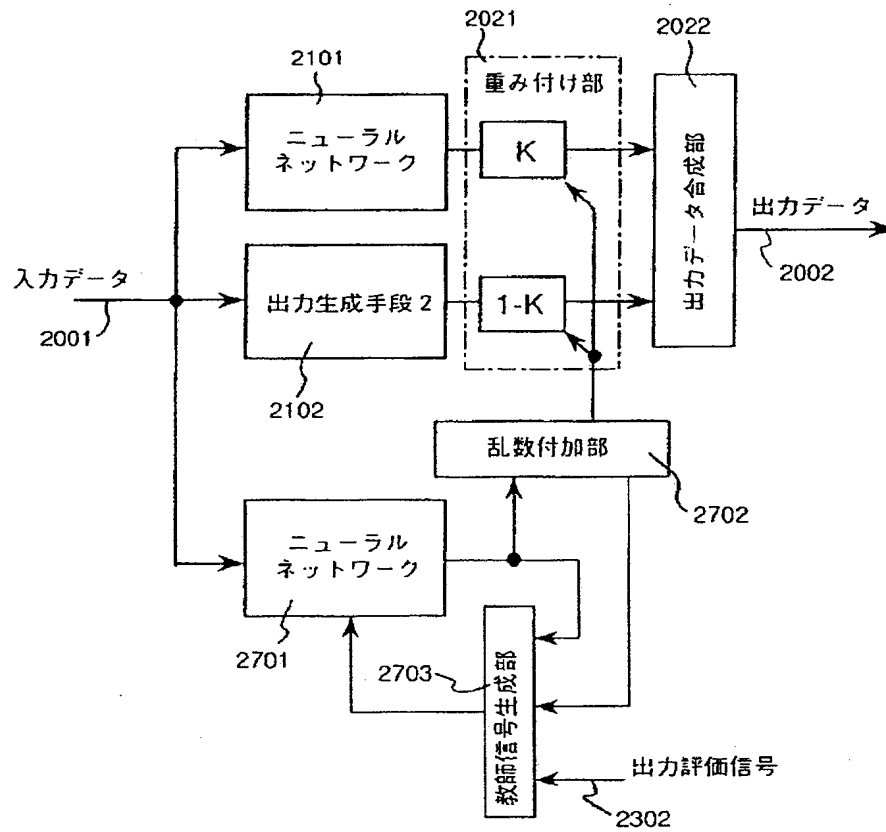
【図48】

図 48

実行分担表			
マイコンピュータ	タスク	負荷率 (%)	内部負荷順位
DC1. PR1	OS11, NC1, CM1, T11	72	2
DC1. PR2	OS12, T12, T13, T14	81	4
DC1. PR3	OS13, T15, T16	75	3
DC1. PR4	OS14, T17, T18	57	1
DC2. PR1	OS21, NC2, CM2	63	3
DC2. PR2	OS22, T21, T22	58	1
DC2. PR3	OS23, T23, T24	60	2
DC2. PR4	OS24, T25, T26	75	4
DC3. PR1	OS31, NC3, CM3	58	1
DC3. PR2	OS32, T31, T32	72	3
DC3. PR3	OS33, T33, T34, T35	60	2
DC3. PR4	OS34, T36, T37	74	4
.	以下省略	.	.

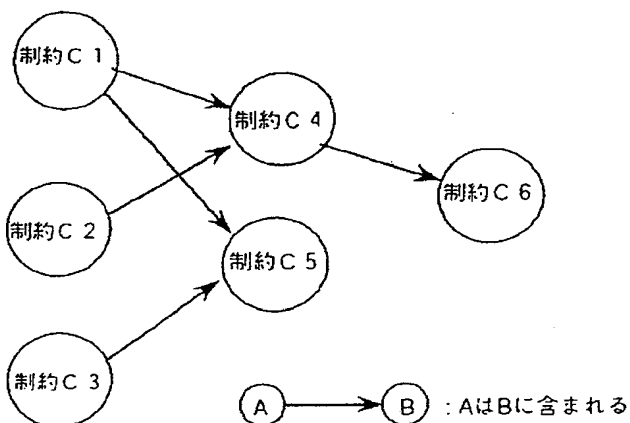
【図 33】

図 33



【図 42】

図 42



【図 53】

図 53

受付用タスクキュー				
タスク	負荷率 (%)	属性	原籍	選択結果
OS13	5	fixed	DC1, PR3	選択
T11	13	communicative	DC1	選択
T16	41	communicative	DC1	選択
T18	26	somewhere	—	非選択
T15	28	somewhere	—	選択

【図 35】

図 35

	コントローラ1	コントローラ2	コントローラ3	コントローラ4
入力変数	X1, X2	X3, X4	X3, X5	X3, X6
出力変数	X3	X5, X6	X7	X8, X9
制約条件	C1: E(X3, X1+X2, e1)	C2: E(X5, X3+X4, e2) C3: E(X6, X3, e3) C4: X5 < 10	C5: E(X7, X3 * X5, e5) C6: X7 < 20	C7: E(X9, X3-X6+X5, e7) C8: E(X8, X3*X4, e8) C9: X8 < 5
制御関数	T1: X3 = X1+X2	T2: X5=X3+X4 T3: X6 = X3	T4: X7=X3*X5	T5: X9= X3-X6+ X5 T6: X8=X3*X4

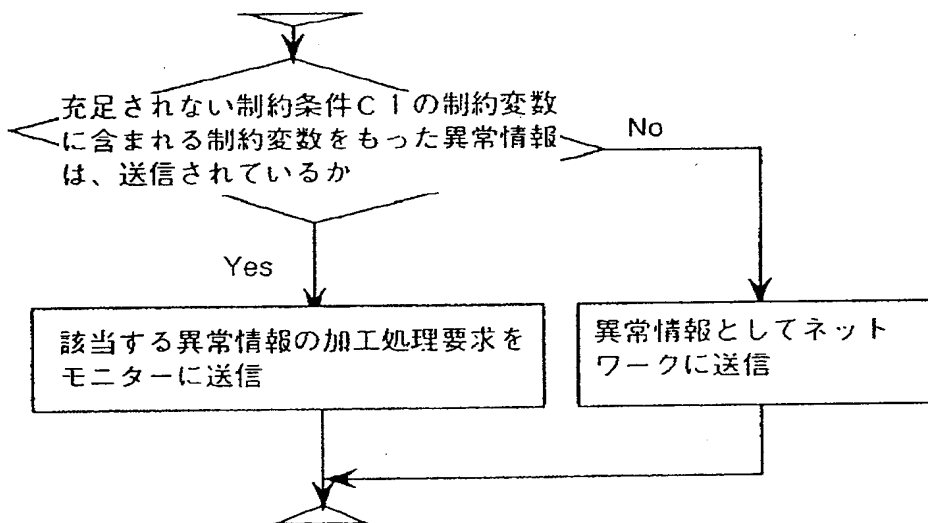
【図 55】

実行分担表			
マイコンビュータ	タスク	負荷率 (%)	内部負荷順位
DC1.PR1	OS12, T12, T13, T14	81	1
DC1.PR2	OS13, T15, T16, T17	88	3
DC1.PR3	OS14, NC1, CM1, T17	84	2
DC2.PR1	OS21, NC2, CM2	63	3
DC2.PR2	OS22, T21, T22	58	1
DC2.PR3	OS23, T23, T24	60	2
DC2.PR4	OS24, T25, T26	75	4
DC3.PR1	OS31, NC3, CM3	58	1
DC3.PR2	OS32, T31, T32	72	3
DC3.PR3	OS33, T33, T34, T35	60	2
DC3.PR4	OS34, T36, T37	74	4
以下省略			

左記の様に網掛けした部分が更新されている

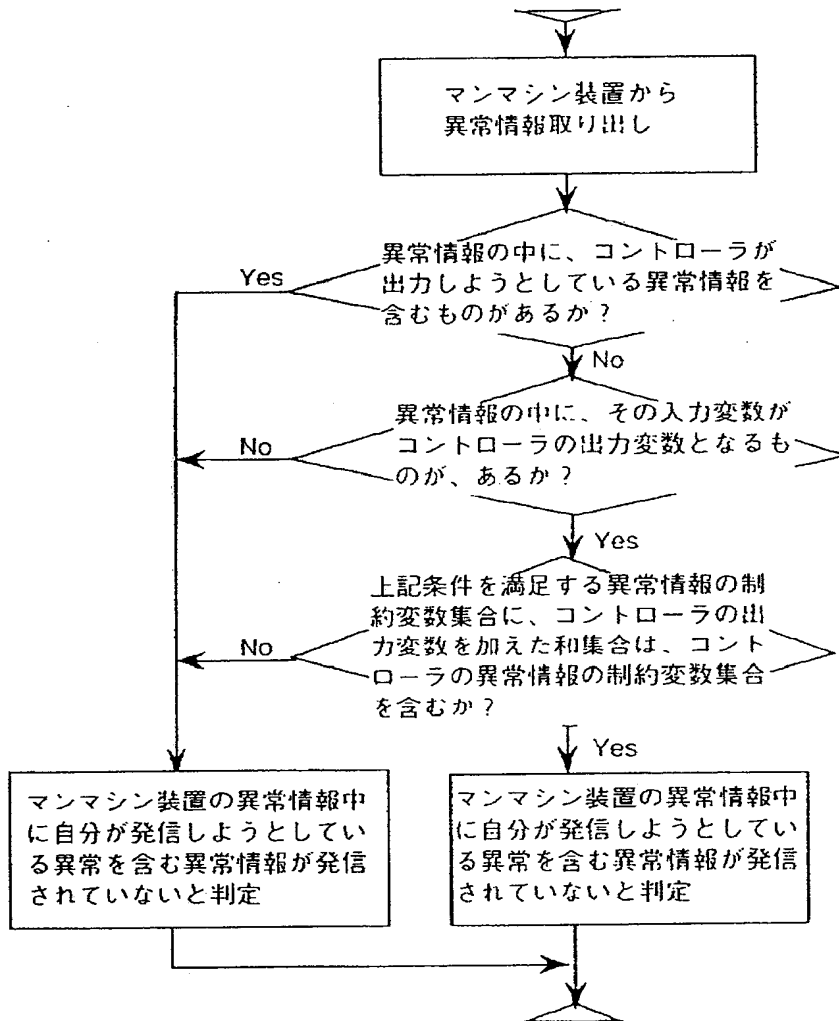
【図 38】

図 38



【図 39】

図 39



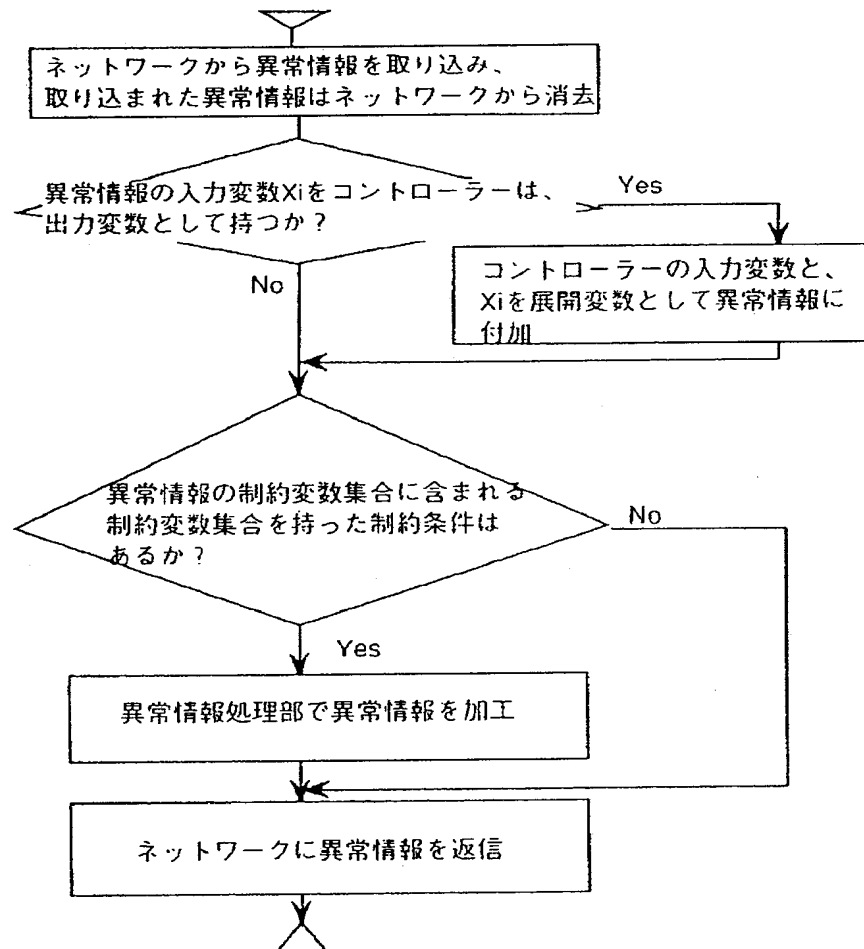
【図 49】

図 49

タスク	負荷率 (%)	属性	原籍
NC1	26	internally	DC1.PR1
CM1	28	internally	OC1.PR1
OS11	5	fixed	DC1.PR1
OS12	5	fixed	DC1.PR2
OS13	5	fixed	DC1.PR3
OS14	5	fixed	DC1.PR4
T11	13	communicative	DC1.PR1
T12	24	somewhere	DC1.PR2
T13	31	communicative	DC1.PR2
T14	21	somewhere	DC1.PR2
T15	28	somewhere	DC1.PR3
T16	41	communicative	DC1.PR3
T17	25	communicative	DC1.PR4
T18	26	somewhere	DC1.PR4
NC2	27	internally	DC2.PR1
-	-	中略	-
OS22	5	fixed	DC2.PR2
OS23	5	fixed	DC2.PR3
-	-	中略	-
T21	18	communicative	DC2.PR2
T22	35	somewhere	DC2.PR2
T23	23	communicative	DC2.PR3
T24	32	somewhere	DC2.PR3
-	-	中略	-
NC3	27	internally	DC3.PR1
CM3	26	internally	DC3.PR1
OS31	5	fixed	DC3.PR1
-	-	以下省略	-

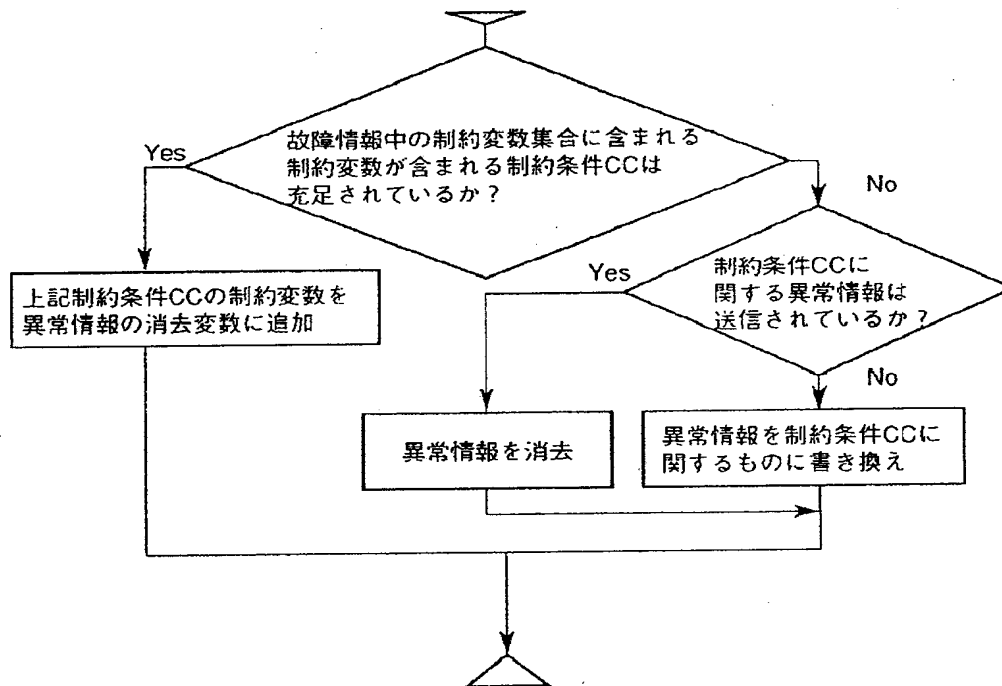
【図40】

図 40



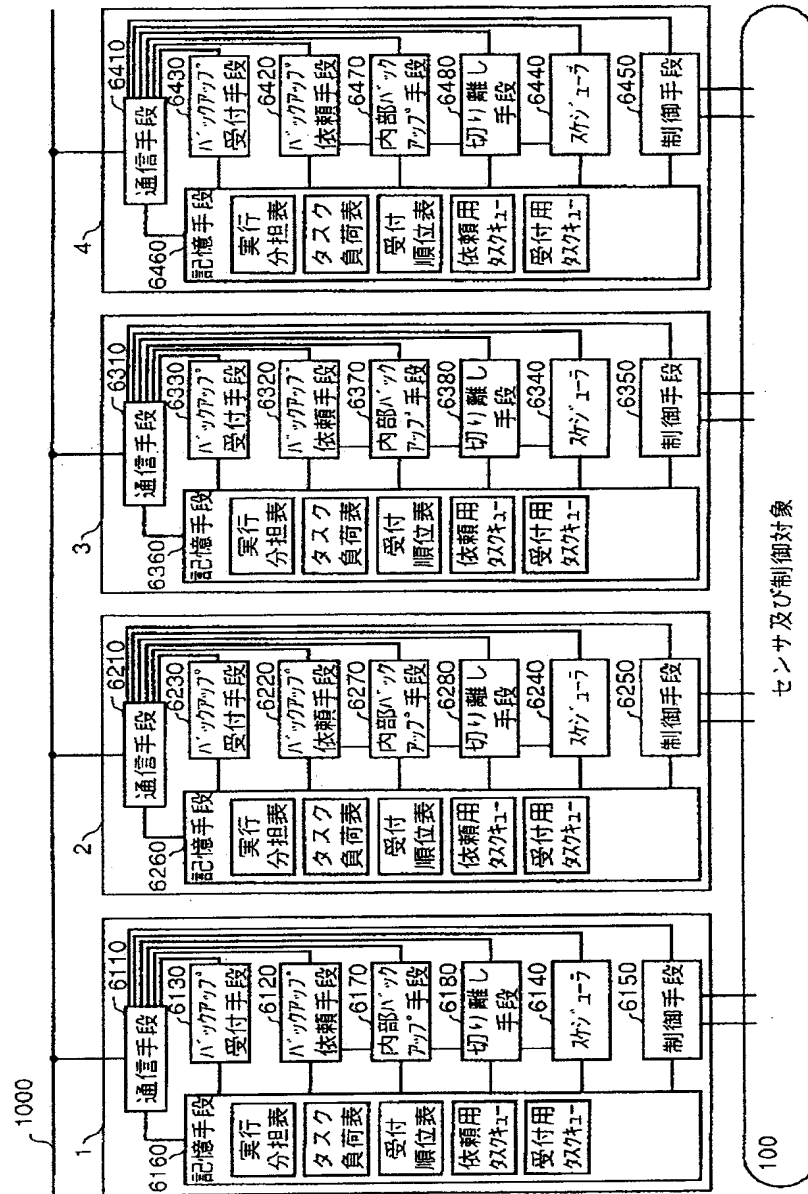
【図41】

図 41



【図43】

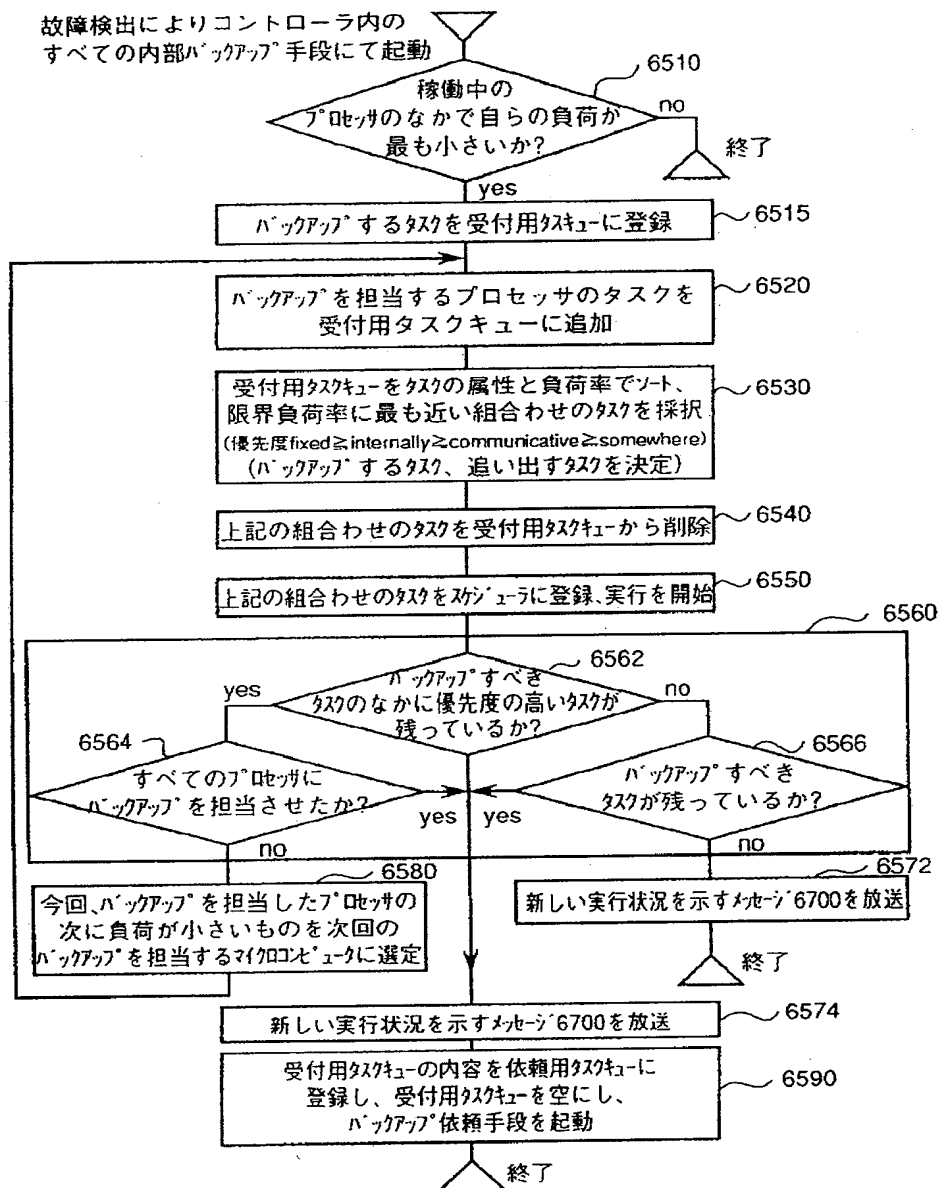
図 43



センサ及び制御対象

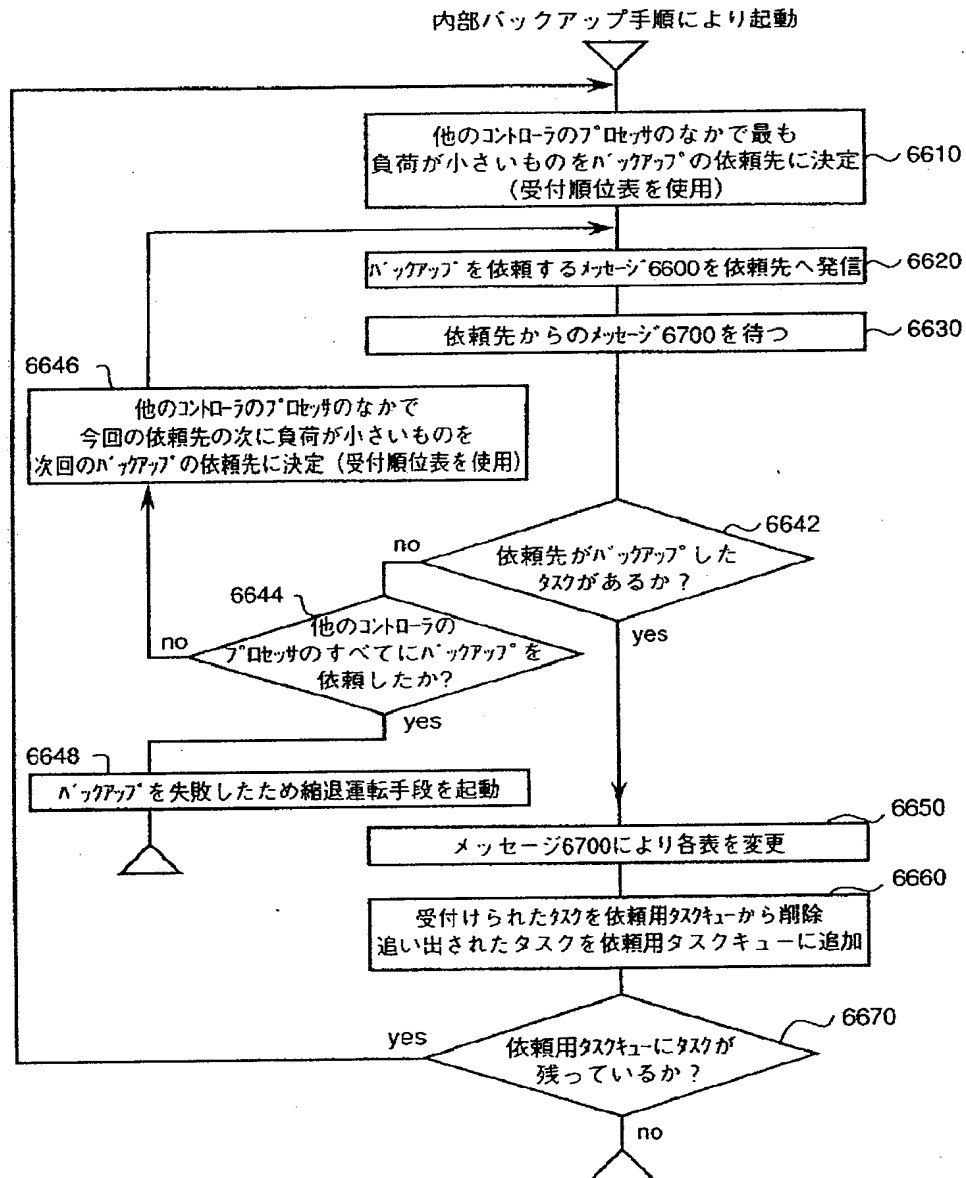
【図44】

図 44



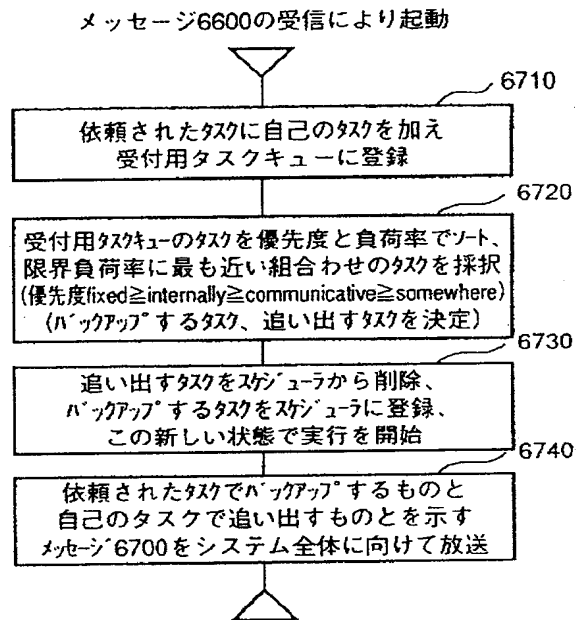
【図45】

図 45



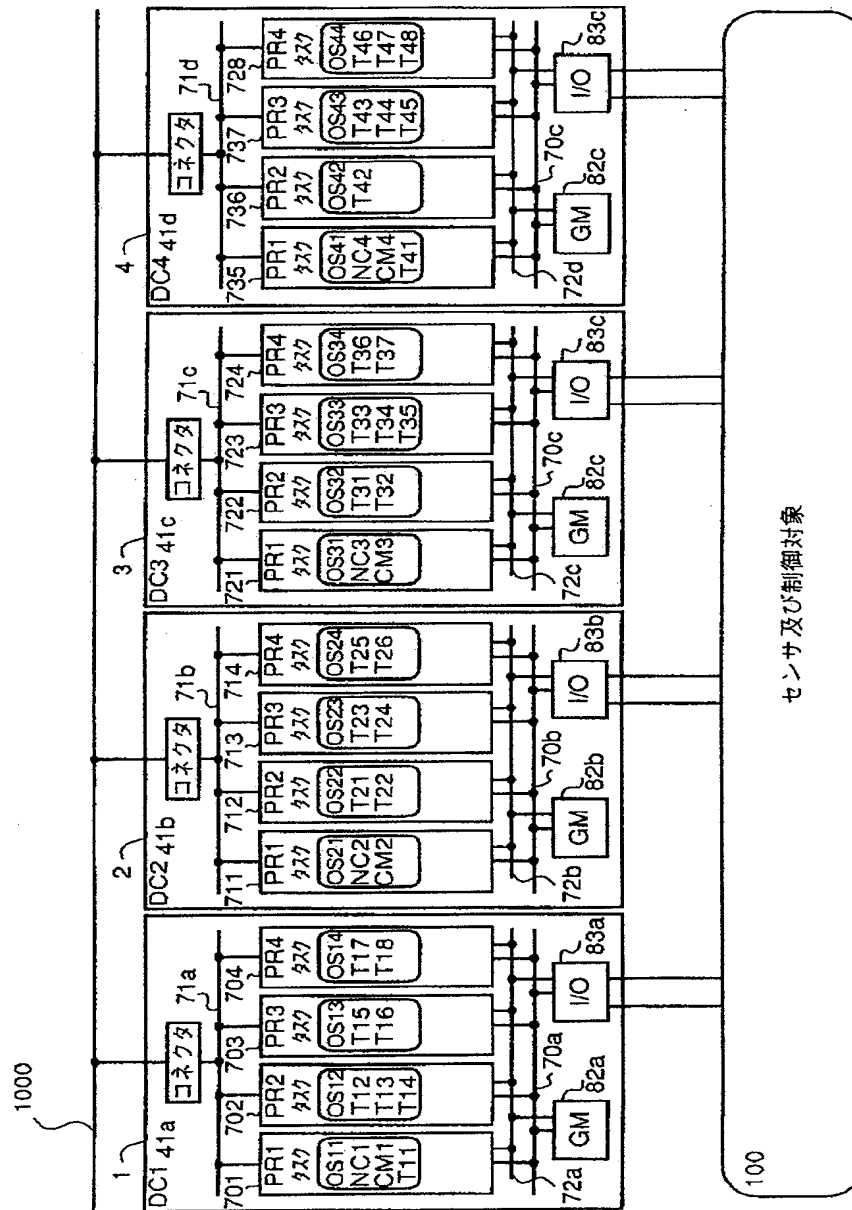
【図46】

図 46



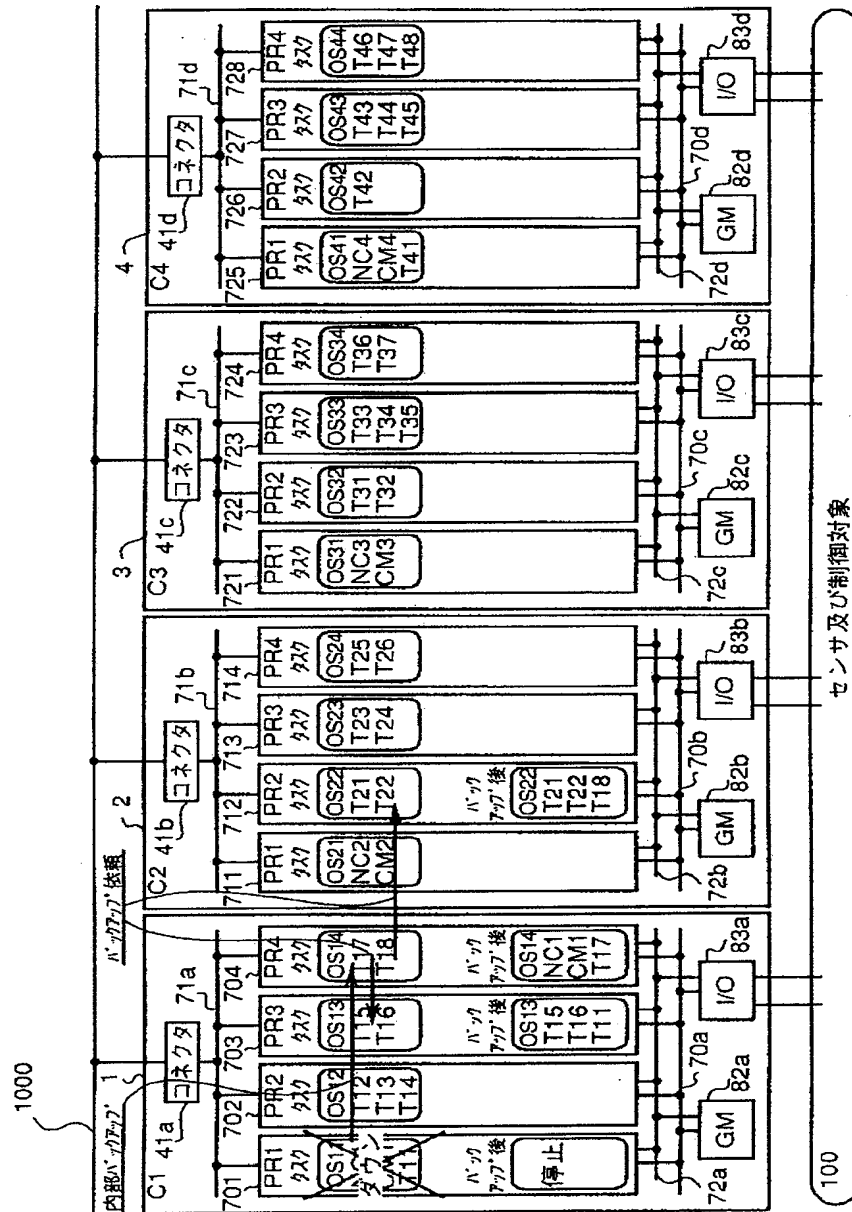
【図47】

図 47



【図51】

図 51



フロントページの続き

(72)発明者 柴田 克成
 東京都国分寺市東恋ヶ窪一丁目280番地
 株式会社日立製作所中央研究所内

(72)発明者 松原 清
 東京都小平市上水本町五丁目20番1号 株
 式会社日立製作所半導体事業部内

(72)発明者 諸岡 泰男

茨城県日立市大みか町七丁目 1 番 1 号 株
式会社日立製作所日立研究所内

(72)発明者 船橋 誠壽

神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所システム開発研究所内

【公報種別】特許法第 17 条の 2 の規定による補正の掲載
 【部門区分】第 6 部門第 3 区分
 【発行日】平成 11 年（1999）11 月 30 日

【公開番号】特開平 8-202409
 【公開日】平成 8 年（1996）8 月 9 日
 【年通号数】公開特許公報 8-2025
 【出願番号】特願平 7-12293
 【国際特許分類第 6 版】

G05B 15/02
 G06F 15/16 370
 470

【F I】

G05B 15/02 M
 G06F 15/16 370 Z
 470 B

【手続補正書】

【提出日】平成 11 年 3 月 31 日

【手続補正 1】

【補正対象書類名】明細書

【補正対象項目名】特許請求の範囲

【補正方法】変更

【補正内容】

【特許請求の範囲】

【請求項 1】複数のプロセッサからなる制御システム用コントローラにおいて各コントローラ内の複数のプロセッサは自己の健全／異常の判別手段、プロセッサの負荷を測定または予測する手段を有し、一つのプロセッサが異常または負荷オーバーとなることを予測または検知した場合、他のプロセッサが負荷の代替を行うことを特徴とするコントローラ。

【請求項 2】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて各コントローラのうち少なくとも一台のコントローラはメモリまたは入出力を共有する複数のプロセッサから構成され、該伝送手段は該コントローラ内プロセッサ群の正常、異常に関係なく該コントローラの上記メモリまたは入出力の内容を読み書きできるように構成したことを特徴とする分散制御システム。

【請求項 3】複数のプロセッサを有し、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの故障を検出する第 1 の手段と、各プロセッサの負荷状況を記憶する第 2 の手段と、各プロセッサの負荷を制御する第 3 の手段を有し、第 1 の手段が一つのプロセッサの故障を検出した場合に、第 2 の手段に記憶された各プロセッサの負荷状況に応じて第 3 の手段が故障したプロセッサのタスクを他の一つ以上のプロセッサに対して選択して代替させることを特徴とするコントローラ。

【請求項 4】複数のプロセッサを有し、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの負荷を検出する第 1 の手段と、各プロセッサの負荷状況を記憶する第 2 の手段と、各プロセッサの負荷を制御する第 3 の手段を有し、第 1 の手段が一つのプロセッサの負荷オーバを検出した場合に、第 2 の手段に記憶された各プロセッサの負荷状況に基づいて、第 3 の手段が負荷オーバとなったプロセッサのタスクと他の一つ以上のプロセッサが実行しているタスクとからその一つ以上のプロセッサの各々が実行するタスクを選択し、それらプロセッサが選択されたタスクを実行することを特徴とするコントローラ。

【請求項 5】複数のコントローラと各コントローラを接続するネットワークを有する分散制御システムにおいて、各コントローラは自コントローラの負荷を測定し測定した負荷をネットワークに送信する第 1 の手段と、任意のコントローラの負荷の代替を他のコントローラに依頼する第 2 の手段と、他のコントローラから代替の依頼に対して自己の負荷状況に基づき回答する手段をする第 3 の手段を有し、何れかのコントローラが負荷オーバあるいは故障した場合、その負荷オーバあるいは故障を検出したコントローラの第 2 の手段は各第 1 の手段から伝えられた負荷状況に応じて負荷の代替を依頼するコントローラを定め負荷の代替を依頼し、依頼されたコントローラの第 3 の手段は自己の負荷状況に応じて代替する負荷を定めこの負荷の実行を制御すると共にこの代替状況を回答することを特徴とした分散制御システム。

【請求項 6】複数のコントローラを有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、該複数のプロセッサ各々は自己の健全／異常の判別手段、各プロセッサの負荷を測定する手段を有し、一つのプロセッサが負荷オーバーになる

と、他のプロセッサのうち最も負荷の小さいプロセッサが負荷の代替を行うことを特徴とする分散制御システム。

【請求項 7】 複数のコントローラをネットワークで接続して制御を実行する分散制御システムにおいて、該複数

のコントローラにおいて感知した異常をネットワーク上に送信するか否かの判定をネットワークからの受信情報に基づいて各コントローラで実行する分散型コントローラの診断方式。